MASTER EN SCIENCES INFORMATIQUES

Comparaison d'outils d'ingénierie des exigences dirigée par les modèles

Van Kerckhoven, M

Award date:
2012

Awarding institution:
Universite de Namur

Facultés Universitaires Notre Dame de la Paix
Faculté d'informatique
Année académique 2011-2012

Comparaison d'outils d'IE dirigée par les Modèles

Michaël van Kerckhoven

Mémoire présenté

en vue de l'obtention du grade de Master en Sciences Informatiques

Promoteurs :
P. Heymans
G. Saval

Facultés Universitaires Notre Dame de la Paix
Faculté d'informatique
Année académique 2011-2012

Comparaison d'outils d'IE dirigée par les Modèles

Michaël van Kerckhoven

Mémoire présenté

en vue de l'obtention du grade de Master en Sciences Informatiques

Promoteurs :
P. Heymans
G. Saval

Pour Elisabeth,

Ma tendre épouse, sans qui je n'aurais eu la force d'entreprendre ce master.

CONTENU

TABLE DES MATIÈRES

TABLE DES FIGURES[1]

[1] Toutes les images issues de *Microsoft® Press®* sont reproduites avec l'autorisation de *Microsoft® Press®* dans un contexte académique et de non-profit.

INDEX DES TABLEAUX

REMERCIEMENTS

Je tiens tout d'abord à remercier Elisabeth qui n'a pas pu profiter de ma pleine présence pendant ses précieuses vacances d'été, et qui a dû subir mon caractère pendant que j'écrivais. Merci également à elle pour la relecture qu'elle a réalisée avec patience.

Ensuite, je tiens à remercier Patrick Heymans pour l'attribution du choix de ce mémoire qui a permis d'approfondir mes connaissances en la matière.

Je remercie également Germain Saval qui a pu me donner des pistes intéressantes grâce à ses connaissances dans le domaine des frameworks de comparaison d'outils d'ingénierie des exigences. Il a pu également m'orienter pour améliorer l'agencement du travail écrit et je l'en remercie.

Sont à remercier également mes parents et beaux-parents sans qui la poursuite de ces études n'aurait pas été possible.

Merci à Ignace Martin qui a pu, en faisant un lien entre la théorie et la pratique, m'éclairer sur la pertinence de certaines parties de mon travail.

Il m'est cher de remercier Benjamine Lurquin, notre secrétaire d'horaire décalé, dont le travail est indispensable, et qui a sympathiquement répondu à mes nombreuses questions.

Je n'oublie pas Pierre Wampach et Benoît Muller qui ont généreusement appuyé et permis l'entreprise de ces études.

Enfin, je remercie Philippe Pâques qui m'a fourni de nombreux ouvrages en la matière.

INTRODUCTION

Il est souvent remarqué, dans les résumés d'études de cas, que la majorité des projets sont réalisés sans outil spécifique à l'ingénierie des exigences[1], ou « IE ». Ce phénomène pourrait s'expliquer de deux manières. La première est que les analystes ne se rendent pas bien compte de la nécessité de cet outil, et sont satisfaits ou habitués de leur utilisation d'une simple « suite bureautique ». La seconde raison est le manque de fonds, de temps ou de connaissances pour la mise en place d'une réelle évaluation impliquant la définition de spécifications d'une part, et l'étude des logiciels COTS (« *Commercial-Off-The-Shelf* ») d'autre part.

Pour cette évaluation, il existe une bonne quantité de bases de données de fonctionnalités qu'un utilisateur peut exiger d'un outil d'IE. L'aperçu que nous en ferons dans le présent travail, montrera que ces listes de fonctionnalités relevées par les chercheurs sont richement fournies. Toutefois, elles ont un manque crucial au niveau des aptitudes des outils en ce qui concerne les modèles visuels. Cette absence a éveillé notre intérêt en suscitant le questionnement initial de notre mémoire. Celui-ci peut être formulé par la question suivante :

« Comment compléter les approches de comparaison d'outils au niveau de l'utilisation des modèles en ingénierie des exigences ? »

Dans notre contexte, un modèle est une représentation abstraite et visuelle d'informations[2] et dont la syntaxe et la sémantique sont définies par un standard. Une exigence vise toute information nécessaire à l'implémentation d'une solution logicielle[3]. L'ingénierie des exigences vise l'ensemble des activités du cycle de vie d'un projet au cours desquelles doivent être acquises et comprises toutes les fonctionnalités et attributs attendus d'un produit[4].

Un ensemble plus complet de concepts seront définis dans le premier chapitre. Ils poseront les jalons théoriques nécessaires au développement de notre étude. Ces concepts permettront également de classifier les exigences et de distinguer les activités de l'IE.

Nous avons été intéressé en premier de savoir quels étaient les modèles disponibles et s'il en existait des variantes. Les différents modèles rencontrés dans la littérature compulsée, qu'ils soient couramment utilisés ou trop peu connus, seront développés et rassemblés dans des catégories en fonction de la similitude de leur utilisabilité. Cette catégorisation aura pour but d'identifier les modèles dans la suite du travail.

Les deux chapitres suivants seront dirigés par l'interrogation sur les apports des modèles en IE et sur la manière de les sélectionner en fonction de l'utilité recherchée. Ils seront une tentative d'aider l'analyste intéressé dans le choix d'un modèle qui lui corresponde. Les modèles seront mis en rapport avec les classes d'exigences étudiées au chapitre premier et la pertinence de chacun d'eux au sein des activités de l'IE sera évaluée.

Le dernier chapitre complétera ces différentes classifications par une liste de fonctionnalités qui peuvent être attendues d'un outil quant aux facilités offertes pour la modélisation des exigences.

[1] (Matulevicius, 2005)
[2] (Podeswa, 2009)
[3] (Beatty, et al., 2012)
[4] (Wiegers, 2003)

CHAPITRE I.

CONCEPTS

1. Classification des exigences

Dans le but d'opérer une comparaison des différents modèles utilisables dans l'IE, nous allons proposer une classification des exigences en fonction de deux critères. Ces derniers, tous deux biens connus dans notre domaine d'étude, sont le degré de fonctionnalité et le niveau d'abstraction des exigences.

Au CHAPITRE IV. , nous créerons une troisième catégorisation abordant une comparaison supplémentaire pertinente des apports des modèles. Nous étudions la nature de ces apports au CHAPITRE III.

1.1. Selon le degré de fonctionnalité

1.1.1. Exigences Fonctionnelles

Ce premier type d'exigences définit ce que le système à construire doit offrir[1]. Il reprend les fonctionnalités logicielles que le produit doit implémenter pour permettre aux utilisateurs d'accomplir leurs tâches[2]. Par cette appellation, nous visons également les fonctionnalités du produit en termes de manipulation de données. Enfin, le domaine d'activité et la portée du produit sont incorporés dans la définition des exigences fonctionnelles , à savoir , pour le premier, ce qu'il doit réaliser et ce dont il doit s'abstenir de produire en regard des systèmes adjacents existants[3].

1.1.2. Exigences Non-fonctionnelles

En ce qui concerne les exigences non fonctionnelles, il est ardu de les déterminer en délimitant leurs caractéristiques à cause des nombreuses catégories existantes ayant chacune leur propre définition. Néanmoins, elles font majoritairement référence aux propriétés qualitatives du produit[4], telles que l'aspect, la convivialité, les besoins de performance ou de sécurité, les normes d'accessibilité, culturelles ou légales, etc.

Dans la littérature traitant de l'ingénierie des exigences, nous trouvons souvent la notion de contrainte définie comme un troisième degré de fonctionnalité[5]. Certains citent des exemples concrets tels que des restrictions sur le système comme son design et son architecture[6] ou encore des conventions d'entreprise à respecter et un vocabulaire à utiliser[7]. Néanmoins, nous ne parvenons que très difficilement à établir une limite entre les contraintes et les exigences. Pour cette raison, nous avons choisi de ne pas définir de troisième classe dans cette classification.

[1] (Pohl, et al., 2011)
[2] (Wiegers, 2003)
[3] (Robertson, et al., 2006)
[4] (Robertson, et al., 2006)
[5] Entre autres dans (Pohl, et al., 2011)
[6] (Kerzazi, 2010)
[7] (Robertson, et al., 2006)

1.2. Selon le niveau d'absraction

1.2.1. Exigences métier

Les besoins du métier sont appelés « exigences métier » et font référence à tout objectif issu du domaine d'activité. Ces exigences indiquent le problème auquel le produit est appelé à répondre tout en en s'abstenant de décrire précisément les tâches qu'il doit accomplir[1]. Le terme métier renvoie dans ce contexte au client ou à l'entreprise demandeuse, mais les exigences métier peuvent également provenir d'un sponsor, d'un département spécifique (de direction ou commercial), ou encore de l'inventeur du produit[2].

Nous pourrions résumer ces exigences par la question « Pourquoi doit-on réaliser le produit ? ».

1.2.2. Exigences utilisateur

A ce niveau, il s'agit de décrire ce que le produit doit implémenter pour que ses utilisateurs puissent remplir les besoins métier. En d'autres termes, elles sont les exigences que la solution logicielle doit satisfaire, expliquées d'un point de vue des utilisateurs finaux. De manière simplifiée, elles désignent les interactions et les réactions du système face à eux[3].

Nous pourrions résumer ces exigences par la question « Quelles tâches le produit doit-il accomplir ? ».

1.2.3. Exigences système

Les exigences système sont parfois appelées « exigences du produit », et désignent ce que le système doit produire pour remplir les exigences utilisateur. Elles définissent les exigences de haut-niveau d'un système contenant lui-même différents sous-systèmes[4].

Nous pourrions résumer ces exigences par la question « Qu'est-ce que le produit doit faire pour réaliser les tâches des utilisateurs ? ».

[1] (Podeswa, 2009)
[2] (Wiegers, 2003)
[3] (Podeswa, 2009)
[4] (Wiegers, 2003)

2. L'évolution des exigences au cours du cycle de vie du projet

D'aucuns pourraient croire que l'ingénierie des exigences ne consiste qu'en la phase de démarrage du projet. Cette idée simpliste laisserait croire qu'une fois les exigences déterminées, il ne resterait plus qu'à construire la solution qui permettrait de les mettre en œuvre. Cette affirmation est uniquement défendable, selon nous, dans les cas des projets de petites tailles.

Cependant, dès que le nombre de parties prenantes augmente ou que le produit se complexifie, les exigences restent continuellement au centre du cycle de vie du projet et se développent selon un véritable processus d'évolution et de validation.

Nous avons tenté, en analysant et recoupant plusieurs[1] classifications de visions différentes, de délimiter ces phases d'évolution des exigences tout au long de l'avancement du projet. Bien que chaque phase fasse l'objet d'un chapitre numéroté, il est évident que ce long processus d'ingénierie des exigences n'est pas linéaire et que des chemins alternatifs ou non programmés peuvent être pris en fonction de certains choix de gestion du projet.

Ainsi, l'idée de devoir acquérir toutes les exigences d'une phase avant de passer à l'étape suivante est un leurre[2]. La Figure I-1 est une tentative de représentation graphique des enchaînements possibles des phases du projet, du point de vue des exigences.

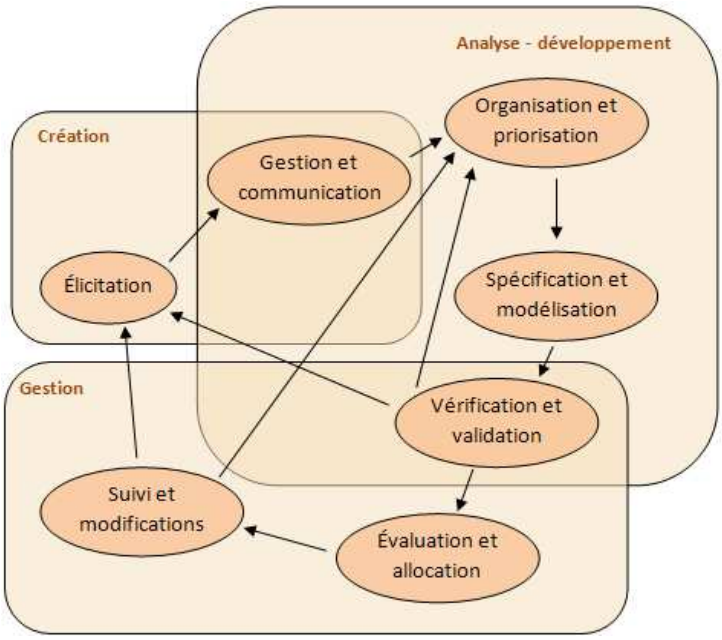

Figure I-1 - Le processus d'évolution des exigences

[1] Les classifications en question sont consultables dans (Robertson, et al., 2006), (International Institute of Business Analysis, 2009) et (Institute of Electrical and Electronics Engineers, Inc., 1998).
[2] (Robertson, et al., 2006)

Le détail complet des différentes phases du cycle de vie du projet sera développé au CHAPITRE V. Toutefois, ci-dessous nous annonçons brièvement les différentes tâches à effectuer au sein de chacune des phases de l'IE :

2.1. Elicitation

Identifier les parties prenantes, la portée du projet (solution scope) et découvrir les exigences.

Définir ce que le produit implémentera, ou lors d'un cycle itératif, définir quelles fonctionnalités une itération fournira.

2.2. Gestion et communication

Gérer un consensus d'implémentation des exigences en fonction de la portée du projet.

Gérer les conflits et faciliter la communication entre les parties prenantes concernées.

Préparer les exigences pour la traçabilité, la réutilisation et le packaging.

Permettre à toutes les parties prenantes d'obtenir une parfaite compréhension des exigences.

2.3. Organisation et priorisation

Déterminer l'importance relative des exigences et s'assurer que les efforts se concentrent sur les plus critiques.

Articuler les exigences autour de leur niveau d'abstraction (exigences du métier, exigences des utilisateurs, exigences du système).

2.4. Spécification et modélisation

Analyser le fonctionnement du métier en s'appuyant sur les modèles, tenter d'apporter des améliorations au système existant en les utilisant pour détecter des problèmes.

Représenter les besoins des parties prenantes par le biais de spécifications formelles, semi-formelles ou non-formelles.

2.5. Vérification et validation

S'assurer du respect d'un standard de qualité supportant une utilisation efficace des exigences dans la suite du projet.

Vérifier qu'elles apportent une valeur au business et que celui-ci en tire un bénéfice, qu'elles remplissent les objectifs métier et vérifient les besoins des parties prenantes.

2.6. Evaluation et allocation

Evaluer la solution afin de déterminer si elle vérifie les exigences des parties prenantes et qu'elle apporte une valeur ajoutée à leur activité.

Séparer les exigences en composants de solutions et « releases » afin de maximiser les bénéfices et minimiser les coûts.

2.7. Suivi et modification

Gérer les cas de tests et s'assurer que le comportement du système corresponde aux exigences.

Découvrir les sources des problèmes identifiés.

Gérer les demandes de changements à apporter au système et estimer leur impact.

CHAPITRE II.

ÉTAT DE L'ART DE COMPARAISON D'OUTILS

1. Ouvrages généraux

En initiant notre étude par la lecture d'ouvrages généraux sur le thème des exigences, nous avons pu prendre connaissance de beaucoup de conseils d'experts en la matière.

Lesdits ouvrages étudiés suivent généralement une ligne conductrice semblable. Ils tentent de parcourir l'ensemble du processus des exigences en énonçant et en expliquant pas à pas les étapes du développement du projet. Plus spécifiquement, ils explicitent les différentes techniques convenant à chacune de ces étapes[1], ou a contrario, celles à éviter. En mettant en avant leur expérience sur le sujet, les auteurs complètent leurs écrits en annotant des recommandations associées à l'un ou l'autre standard de modélisation[2].

Les outils d'IE occupent fréquemment une place assez restreinte dans la littérature générale d'analyse business, laissant croire que les auteurs ne jugent pas leur utilisation absolument nécessaire. Pourtant, bon nombre d'entre eux soulignent – cela fait évidemment un argument intéressant dans les introductions – qu'une bonne gestion des exigences est essentielle pour la réussite d'un projet.

Nous pensons, comme l'atteste l'entreprise de ce travail, qu'un bon outil est indispensable à la gestion des exigences et qu'il permettra de limiter les erreurs d'analyse, de spécification et de compréhension. En raison du prix généralement élevé dans l'acquisition de celui-ci, une présentation d'une relation entre les coûts de correction d'une erreur d'IE et le moment de sa découverte (comme présenté dans la Figure III-1) permettra facilement de faire comprendre aux parties prenantes l'enjeu financier en question et donc de justifier cette dépense.

1.1. Software Requirements 2nd Edition[3]

Wiegers consacre dans son livre un chapitre entier aux outils d'IE, dans lequel il fournit au lecteur un guide pour mieux choisir son outil en fonction de critères personnalisés. Ainsi, dans cet ouvrage, le lecteur est invité implicitement à lister ses propres critères.

La sélection se fait sur base de neuf étapes à suivre dans l'ordre suivant :

1. Définir les exigences attendues de l'outil d'IE (fonctionnalités, interopérabilité, plateformes disponibles, modes d'accès, ...).
2. Lister une quinzaine de facteurs qui vont être décisifs pour la sélection. Inclure des facteurs subjectifs, mais exclure le coût.
3. Distribuer un total de 100 points de sorte à donner davantage ou moins de poids aux facteurs selon leur degré d'importance.

[1] Le *BABOK® Guide*, tentative de standardisation des bonnes pratiques de l'analyse business, en est un bel exemple (International Institute of Business Analysis, 2009).
[2] Comme c'est le cas par exemple de l'ouvrage *Mastering the Requirements Process* de Suzanne et James Robertson (Robertson, et al., 2006).
[3] (Wiegers, 2003)

4. Lire les descriptions des outils ou regarder des démos afin d'attribuer un score à chaque facteur décisionnel. Estimer les scores des facteurs subjectifs et reporter leur réelle attribution à plus tard, lors du test des outils.
5. Calculer le total de chaque candidat en calculant la somme des scores pondérés par les poids définis à l'étape 3.
6. Solliciter l'avis d'utilisateurs pour chaque candidat afin de suppléer à l'évaluation réalisée et à l'argumentaire de vente du fabricant.
7. Obtenir des copies de test et définir un processus d'évaluation.
8. Evaluer les outils par l'intermédiaire d'un projet réel et ajuster éventuellement les scores.
9. Combiner les totaux pondérés, les coûts (sans oublier d'éventuelles récurrences), les impressions subjectives de l'équipe et les avis externes afin d'effectuer choix.

Bien que cette méthode puisse paraître efficace et simple à mettre en place, nous osons émettre l'avis qu'elle peut facilement mener à un dilemme. En effet, l'auteur reste assez flou sur l'importance à attribuer à chaque poste décisionnel et incorpore beaucoup d'avis subjectifs dans le processus. Lors de la phase finale, le risque causé par l'intégration d'avis extérieurs dans la décision, si ceux-ci sont contraires à l'étude menée, peuvent aboutir à la confrontation suivante : devoir choisir entre l'étude effectuée plutôt que les avis externes en rejetant également ce qui peut être pertinent, ou bien suivre les conseils de son équipe et dans ce cas, jeter en quelque sorte le travail que l'on vient d'effectuer.

1.2. Requirements Engineering Fundamentals[1]

Dans ce livre, *Pohl et Rupp* discutent des moyens possibles pour mesurer l'adéquation d'un outil à gérer l'ingénierie des exigences. Ils proposent une séparation des fonctionnalités sous la forme d'une roue offrant différentes vues sur un outil d'IE. Selon eux, cette approche permet d'évaluer et de prioriser les exigences des outils d'IE.

[1] (Pohl, et al., 2011)

Figure II-1 - Différentes vues sur un outil d'IE (Pohl, et al., 2011)

Comme en témoigne la Figure II-1, leur roue comporte sept vues explicitées ci-dessous. Chaque quartier est voué à soutenir l'utilisateur dans la définition de critères précisant ses attentes de l'outil.

1. Vue « **Projet** » : Possibilités de l'outil en terme de gestion de projet (planning, exécution).
2. Vue « **Utilisateur** » : Interface agréable, connexions simultanées d'utilisateurs, gestion des droits/rôles.
3. Vue « **Produit** » : Fonctionnalités proposées par l'outil.
4. Vue « **Processus** » : Degré de support des méthodes proposées par l'outil pour gérer les activités de l'IE.
5. Vue « **Fournisseur** » : Réputation du produit et du fournisseur, position sur le marché, services de support.
6. Vue « **Technique** » : Exigences logicielles et infrastructurelles pour l'intégration de l'outil, performances, interopérabilité, sécurité, etc.
7. Vue « **Economique** » : Coûts d'acquisition, d'intégration, d'apprentissage, d'opération, de maintenance et de support.

A l'exception de ces points, les auteurs ne donnent pas de critères pour évaluer les outils. Ils ne fournissent pas non plus de méthode de sélection.

2. Projets de recherche spécifiques

Des projets de recherche visant plus particulièrement les outils d'IE et la manière d'en réaliser une bonne comparaison en fonction des besoins d'une entreprise ou d'un projet, ont été également étudiés.

Plus spécifiquement, trois travaux de recherche récents dans le domaine de la comparaison des outils d'IE ont appuyé notre travail. Leur approche est sensiblement différente :

La première est une étude réalisée par *l'INCOSE* donnant à une large gamme d'outils d'IE un indice de conformité en regard d'une liste de fonctionnalités.

La deuxième, réalisée un département de recherche de la société *Seilevel*, et publiée par *Beatty et Ferrari*, comporte une large gamme d'exigences ayant servi à la sélection de leur propre outil d'IE.

La troisième, publication d'une thèse de doctorat de *Matulevicius*, fournit un premier cadre de travail pour l'évaluation des exigences fonctionnelles d'un ou plusieurs outils, ainsi qu'un second pour l'évaluation des exigences non-fonctionnelles. Il fournit également une méthode de sélection.

Il existe également d'autres listes de fonctionnalités ou « frameworks », nous pouvons citer celui de *Volere* ou encore la norme *ISO/IEC TR 24766:2009*.

2.1. INCOSE Requirement Management Tools Survey[1]

Cette étude, publiée dans sa dernière version en Août 2010 par l'*International Council on Systems Engineering*, présente un ensemble de fonctionnalités qui peuvent être attendues d'un outil d'IE. Des questions ouvertes sur le logiciel sont également abordées. Un score (*Full* / *Partial* / *None*) situe l'outil pour chaque fonctionnalité.

Ce formulaire, mis au point par l'*INCOSE Requirements Working Group* (RWG) a été rempli par les éditeurs d'un ensemble de 34 outils d'IE. Ces derniers font partie d'une base de données alimentée depuis les années 1990 par l'*INCOSE Tools Database Working Group* (TDWG), qui depuis quelques années semble avoir cessé d'évoluer[2].

L'étude regroupe les fonctionnalités en un ensemble de douze catégories définies de la manière suivante :

1. Identification et capture des exigences
2. Capture de l'architecture des éléments du système
3. Transfert des exigences
4. Traçabilité
5. Gestion de la configuration

[1] (INCOSE Requirements Working Group, 2010)
[2] (INCOSE Requirements Working Group (2), 2010)

6. Documents et autres rapports de sortie
7. Collecticiel
8. Connexion à d'autres outils
9. Environnement du système
10. Interface utilisateur
11. Standards
12. Support et maintenance
13. Formations
14. Autres commentaires.

L'*INCOSE* apporte une liste assez complète de fonctionnalités pour évaluer les outils, mais ne conseillent pas de méthode pour en sélectionner un en fonction des résultats de l'évaluation.

Il est important de souligner que les réponses ont été apportées par les éditeurs de chaque logiciel, mais que le TDWG, en les parcourant, s'est réservé le droit de modifier toute exagération ou information jugée incorrecte.

Nous ne détaillons pas ici la totalité de l'étude dont les 67 fonctionnalités nécessitent plusieurs pages. La liste complète des outils concernés, ainsi que celle des fonctionnalités faisant l'objet de l'étude, peuvent être consultées dans l'Annexe I.

2.2. Seilevel Requirements Gathering Management Tool Evaluation[1]

Seilevel est une entreprise de consultance spécialisée dans l'ingénierie des exigences, dont un département de recherche a réalisé un première évaluation d'outils en 2007 et une seconde en 2010. L'objectif visé par ces dernières résidait dans le choix d'un nouvel outil à utiliser en interne ou chez leurs clients.

Les études qu'ils ont réalisées ont été rendues publiques dans l'intention délibérée d'aider quiconque désirant effectuer la même recherche.

Leur seconde étude a été élaborée en s'inspirant de la première et en y ajoutant quelques critères tirés de l'étude de l'INCOSE ainsi que des idées émanant des brainstormings réalisés en interne. La différence avec INCOSE se situe dans le fait que leur évaluation se base sur l'appréciation d'une tierce partie et non sur celle des éditeurs eux-mêmes. En outre, ils intègrent un niveau de plus dans leur score d'évaluation.

2.2.1. Critères d'évaluation et calcul du score

Les critères, au nombre de 203 et répartis dans 50 groupes, ne sont pas énumérés ici mais sont consultables dans l'Annexe XVI.

[1] (Beatty, et al., 2011)

Nous énumérons ci-dessous les niveaux de priorité et de conformité de chaque critère proposés par *Beatty et Ferrari*, qui sont organisés sous la forme d'une échelle de scores.

Définitions des priorités des exigences[1]

- 3 – Haute : Fonctionnalité à avoir obligatoirement ;
- 2 – Moyenne : Fonctionnalité intéressante à avoir, fournit une flexibilité ;
- 1 – Basse : Fonctionnalité peu importante mais qui rendrait l'outil plus agréable.

Scores d'évaluation[2]

- 3 : Entièrement supporté par l'outil ;
- 2 : Supporté mais nécessite de petites solutions de contournement ou manque de certaines fonctionnalités ;
- 1 : Peu supporté, nécessitant de grosses solutions de contournement, ou fonctionnalités minimes uniquement ;
- 0 : Non supporté.

Le calcul des scores des outils est réalisé par la somme des scores de chaque exigence pondérés par sa priorité :

$$\text{score}_{\text{outil}} = \sum(\text{score}_{\text{exigence}} * \text{priorité}_{\text{exigence}})$$

Afin de s'assurer de n'être passé à côté d'aucune fonctionnalité, *Seilevel* a demandé aux éditeurs de logiciels de réaliser la même appréciation en utilisant leur système d'attribution de score.

2.2.2. Approche d'évaluation[3]

1. Créer une liste complète d'outils d'IE pour évaluation.
2. Créer une liste priorisée de critères pour les outils.
3. Sélectionner depuis l'ensemble complet des critères une liste plus courte pour une première passe de sélection.
4. Filtrer la liste pour les outils de gestion pure des exigences, en excluant ceux de définition, prototypage ou spécifiques aux projets Agile.
5. Publier les critères d'outils de gestion des exigences et la liste des outils à des fins de révision par l'industrie.
6. Evaluer tous les outils de gestion des exigences d'après la liste de critères de première passe.
7. Evaluer le 15 meilleurs outils issus de l'évaluation de première passe d'après la liste complète des critères.

[1] (Beatty, et al., 2011)
[2] (Beatty, et al., 2011)
[3] (Beatty, et al., 2011)

8. Faire évaluer les 15 meilleurs outils par leurs éditeurs d'après les mêmes critères afin d'identifier d'éventuelles discordances.
9. Publier les résultats de l'évaluation détaillée des outils de gestion des exigences à des fins de révision par l'industrie.
10. Implémenter et évaluer les trois meilleurs outils issus de l'évaluation complète sur des projets réels.
11. Publier les résultats de l'évaluation sur projet des outils de gestion des exigences.

Leur méthode nous semble très complète et a l'avantage d'avoir servi ses auteurs pour une étude interne. Néanmoins, nous estimons qu'elle est difficile à mettre en place lorsqu'il s'agit d'évaluer consécutivement (ou parallèlement) trois outils sur des projets réels.

2.3. RE-Tools Evaluation Approach[1]

La très grande majorité des auteurs débattant du sujet des outils d'IE et de leur comparaison, fait référence au travail de *Matulevicius* et à son « *RE-Tools Evaluation Approach* » (R-TEA), ou « approche d'évaluation des outils d'ingénierie des exigences », si nous pouvons nous permettre une traduction.

Cette approche consiste en deux cadres de travail : le premier mis au point pour l'évaluation des exigences fonctionnelles des outils et le second pour celle des exigences non-fonctionnelles.

Nous allons présenter d'une manière succincte les deux cadres de travail, ou « frameworks », proposés dans la méthode R-TEA. Une description plus exhaustive des activités de ceux-ci est consultable dans l'Annexe XV.

2.3.1. Framework for Evalutation of Functional RE-tool Requirements

Matulevicius propose un cadre de travail réparti en trois groupes, chacun étant découpé en catégories de fonctionnalités ou, selon les termes originaux de l'auteur, en différentes activités que l'outil d'IE devrait pouvoir réaliser. Son schéma est présenté dans la Figure II-2.

Representation dimension

Ce premier groupe évalue les différents moyens offerts par l'outil pour exprimer les exigences via un langage informel, semi-formel et formel. En outre, il discute de la possibilité d'établir des liens entre les différentes représentations d'une même exigence. L'auteur place également dans cette catégorie l'évaluation des connexions (import/export ou communication directe) avec d'autres logiciels.

[1] (Matulevicius, 2005)

Agreement dimension

Ce deuxième groupe cite d'une part, des fonctionnalités essentiellement orientées vers la gestion des utilisateurs avec leur création, leur assignation à des groupes, l'attribution de leurs apports aux exigences, l'approbation de ceux-ci et la possibilité d'un travail coopératif. D'autre part, les exigences enregistrées doivent pouvoir être garnies par les utilisateurs d'un ensemble de propriétés faisant office de filtres ou de tris et facilitant leur recherche et leur regroupement. Enfin, la présence d'une série d'activités permettant la définition des termes est notable car elle propose une compréhension éclairant aussi bien une personne technique qu'une personne non avertie.

Specification dimension

Ce dernier groupe traite de la faisabilité d'une réutilisation des exigences, par exemple au moyen de fonctions d'import ou export vers un « repository ». Il s'agit présentement de mesurer la capacité de l'outil à générer des rapports, de la documentation et des spécifications. La manière de produire ceux-ci est évaluée, dont la conformité à des standards formels de documentation.

Schéma du framework

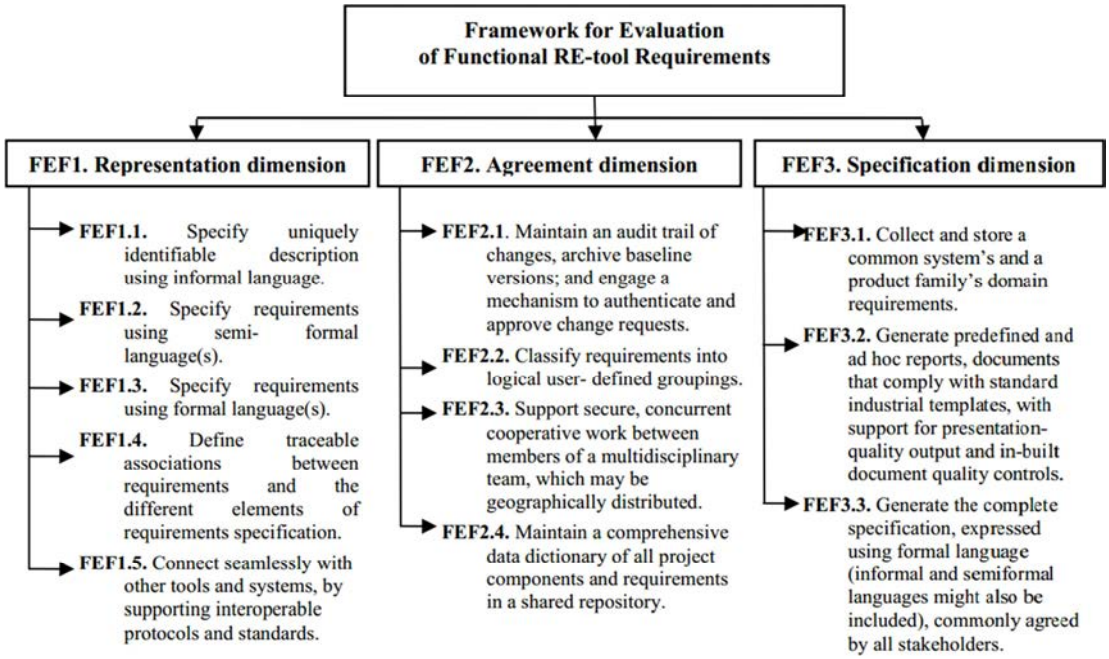

Figure II-2 - Framework for Evaluation of Functional RE-tool Requirements (Matulevicius, 2005)

2.3.2. Framework for Evaluation of Non-Functional RE-tool Requirements

D'une façon similaire au premier cadre de travail, Matulevicius propose une hiérarchisation des caractéristiques non-fonctionnelles d'un outil d'IE

Process requirements

Ce premier groupe énonce un ensemble de catégories de contraintes posées par l'équipe technique ou les analystes eux-mêmes, et la capacité de l'outil à respecter des modèles de processus et différents standards (par exemple : standards de modélisation).

Product requirements

Il s'agit ensuite, dans ce deuxième groupe, de définir les exigences purement non-fonctionnelles de l'outil d'IE telles que l'utilisabilité (utilisation « *user-friendly* », facilité d'apprentissage, compréhension de l'outil, …), la fiabilité (disponibilité, gestion d'erreurs, …), la performance (rapidité, précision, gestion des ressources, …), et la maintenabilité (extensibilité et testabilité, outils de maintenance, …).

External requirements : Organisational requirements

Le troisième groupe est séparé en deux parties dont la première soulève des questions de coûts (ceux liés à l'outil ou à son évaluation), des questions de prise de décision par rapport à l'expérience des utilisateurs, aux contraintes de l'organisation, de la politique en vigueur et des aspects socio-culturels.

External requirements : Requirements for business parties

Dans sa deuxième partie, le troisième groupe évalue les facilités offertes par l'éditeur pour la formation à l'utilisation de l'outil. Le support, son temps de réponse, les services de maintenances et leur disponibilité sont également répertoriés.

Schéma du framework

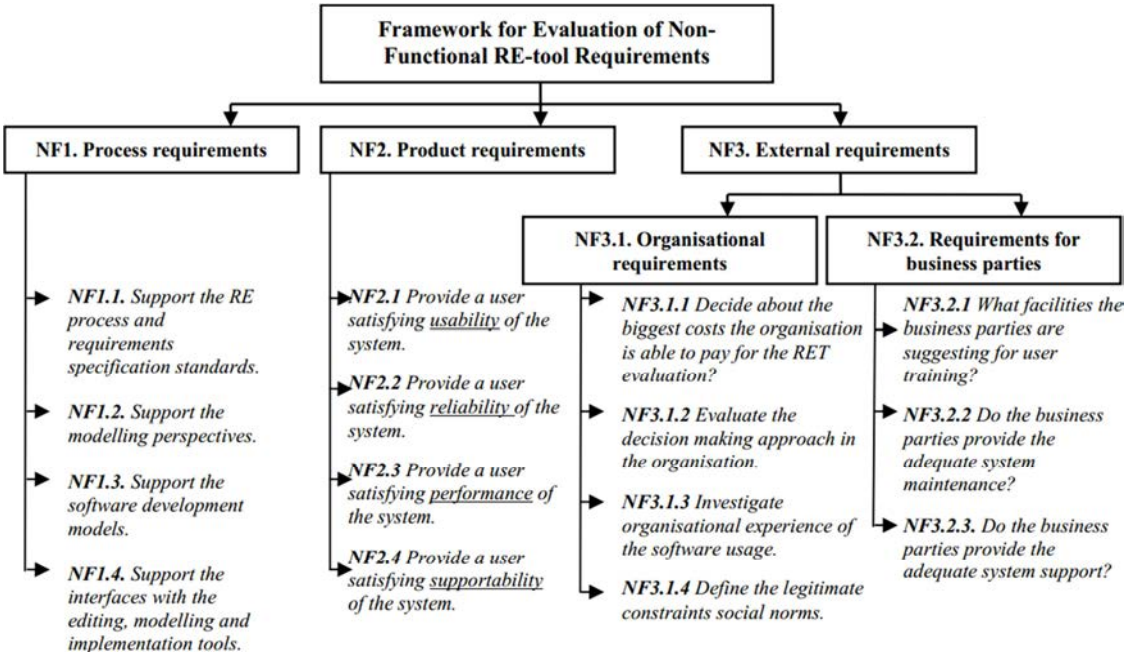

Figure II-3 - Framework for Evaluation of Non-Functional RE-tool Requirements (Matulevicius, 2005)

2.3.3. La méthode R-TEA

Enfin, après avoir défini ses cadres de travail, *Matulevicius* propose, sous la forme d'un diagramme d'états, sa marche à suivre pour l'évaluation des outils et pour la sélection d'un candidat. Cette méthode est décrite dans la Figure II-4.

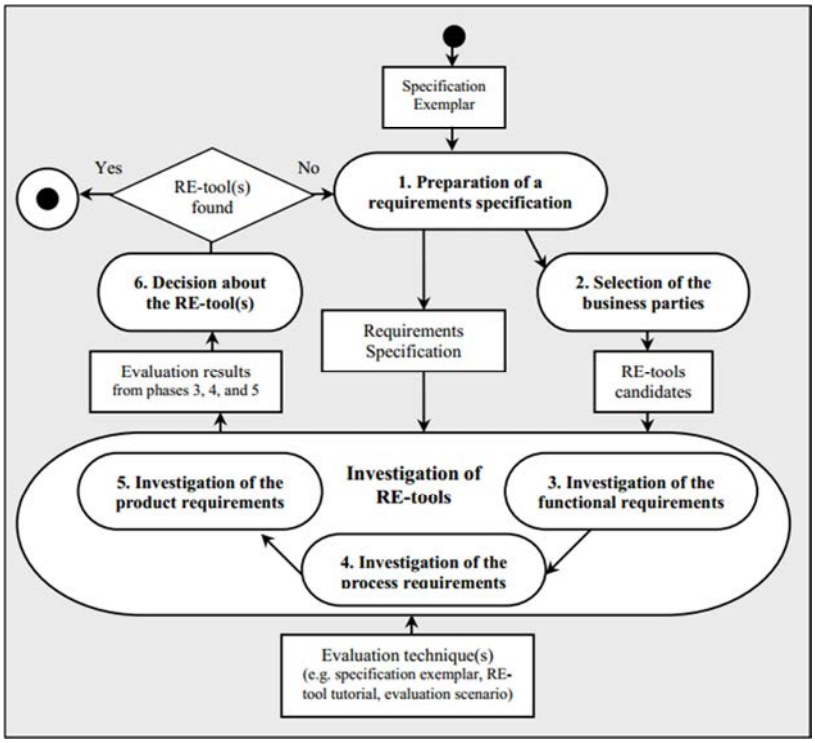

Figure II-4 - RE-Tools Evaluation Approach (Matulevicius, 2005)

La méthodologie implique l'utilisation d'une liste de spécifications appelée par l'auteur « exemplaire de spécification » et calqué sur les deux frameworks de la méthode, que les futurs utilisateurs de l'outil doivent prioriser. Par ce biais, ils pourront exprimer leurs besoins et participer à la sélection de l'outil. Priorisée, cette liste devient la spécification des exigences et sera utilisée dans les phases d'investigation. Lorsque cette évaluation est terminée, une décision est à prendre entre l'adoption du meilleur outil recouvrant certaines habitudes de travail, ou l'adaptation du processus de travail à celui du ou des meilleurs outils sélectionnés. Si aucun d'eux ne satisfait l'utilisateur une fois le processus abouti, la méthode conseille de répéter l'application d'évaluation à partir de la priorisation.

Le processus de sélection est très demandeur de temps si l'on s'attèle à la comparaison au moyen de frameworks soit non adaptés aux outils d'IE, soit ne fournissant pas de méthodologie d'utilisation. L'approche R-TEA construite par *Matulevicius* tente d'aller en ce sens en proposant, en plus de ses deux frameworks spécifiques aux outils d'IE, une sorte de mode d'emploi permettant de diminuer la quantité de temps nécessaire à effectuer cette comparaison.

CHAPITRE III.

ÉTUDE DES MODÈLES

1. Introduction

Importance de la spécification des exigences

Dans le domaine du développement logiciel, il est actuellement reconnu que l'ingénierie des exigences joue un rôle prépondérant dans la réussite d'un projet. Des erreurs d'analyse des exigences se paient très cher si elles ne sont pas découvertes à temps. Cette affirmation est appuyée par la mise en comparaison du coût d'une erreur d'IE par rapport à l'état d'avancement du projet :

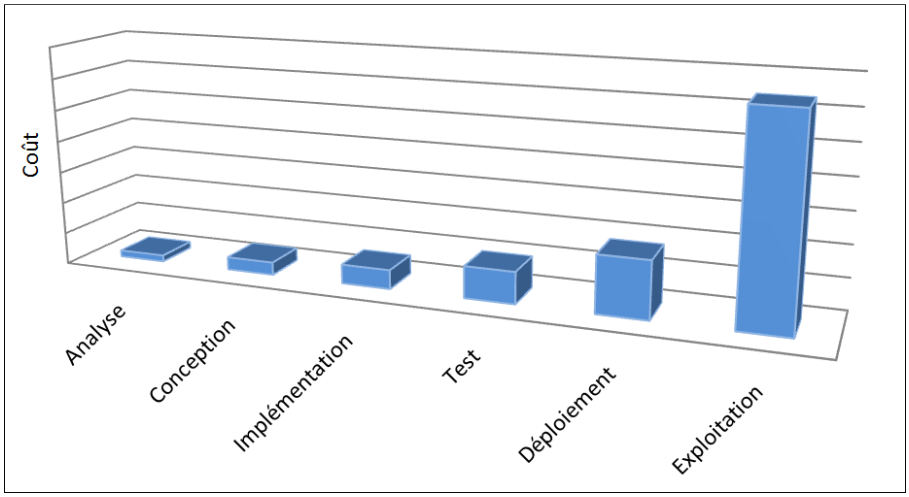

Figure III-1 - Cout relatif de correction des erreurs d'ingénierie des exigences

Les attentes du client doivent être comprises intégralement par l'équipe (gestionnaires de projet, analystes, équipe technique). Ces besoins se traduisent en exigences métier, exigences utilisateur et exigences système.

Ces trois niveaux d'abstraction doivent être analysés et documentés. Les principaux outputs de l'analyse des exigences sont en effet les spécifications qui permettent au client de valider la bonne compréhension de ses attentes (aux trois niveaux) et de vérifier que la solution proposée remplisse les objectifs attendus[1].

Une image vaut mille mots

Les projets de développement logiciel sont souvent blâmés de ne pas fournir de spécifications d'exigences claires, ou encore que celles-ci soient transmises de manière incomplète. Le fait que les exigences soient principalement formulées de manière textuelle, par le biais de phrases commençant par « Le système devra », en est une des causes. Bon nombre de projets se satisfont d'un document d'exigences contenant une liste de ces phrases en guise de spécifications[2].

Nous pensons que la modélisation des exigences peut apporter de grands atouts à une spécification. Si les modèles sont combinés aux spécifications écrites, ils assurent une totale compréhension des

[1] (International Institute of Business Analysis, 2009)
[2] (Beatty, et al., 2012)

exigences ; il s'agit donc de ne pas se limiter aux modèles, ni aux spécifications textuelles[1]. Les modèles apportent une clarté et diminuent les interprétations possibles en raison du formalisme que leur standard définit. Leur principal atout est l'aspect représentatif comme le souligne l'adage bien connu « Une image vaut mille mots ». Ils sont agréables et faciles à lire rapidement car leur syntaxe est plus propice au fonctionnement du cerveau humain que celle des éléments textuels[2]. Certains modèles donnent une vie au document en pouvant être « actionnés » comme des automates.

L'intérêt des modèles

Lorsqu'est prononcé le mot « modélisation », d'aucuns pensent qu'il est uniquement question de descendre dans l'architecture technique et infrastructurelle du système. Cependant, outre cet aspect, les modèles conviennent très bien à l'ingénierie des exigences pour les raisons évoquées précédemment. Ils peuvent y être utilisés pour représenter visuellement différentes perspectives telles que les concepts, les processus ou les interactions, et cela à différents niveaux d'abstraction (métier, utilisateur, système). Nous discuterons plus en détail des apports des modèles au CHAPITRE IV.

La réalisation de ces modèles possède un triple intérêt au sein du cycle de vie du projet. Outre le fait d'aider l'analyste dans ses phases d'élicitation, ces derniers apportent une certaine facilité de vérification des exigences avec les parties prenantes du métier, ainsi qu'une meilleure communication et compréhension de celles-ci par l'équipe technique (développeurs et testeurs)[3]. Nous nous étendrons de manière plus détaillée sur l'intérêt des modèles au sein du cycle de vie du projet dans le CHAPITRE V.

[1] (Wiegers, 2003)
[2] (Buzan, 1976)
[3] (Beatty, et al., 2012)

2. Objets de modélisation

Dans ce point, nous présentons tous les concepts de l'IE que nous avons jugé opportun de modéliser. Ces derniers sont repris sous la forme d'une liste des exigences de modélisation attendues d'un outil d'IE. Pour chaque concept ou objet de modélisation (OM)[1], nous présentons différentes solutions de modélisation par le biais de standards reconnus.

Les standards repris dans ce travail sont les suivants :

- UML : « *Unified Modeling Language* », bien que conçu principalement pour modéliser la conception technique des systèmes, il est également souvent utilisé à plus haut niveau en ingénierie des exigences ;
- RML : « *Requirements Modeling Language* », spécialement conçu pour la modélisation des exigences et inventé par *Anthony Chen*[2] ;
- i* : « *eye-star* » a été mis au point essentiellement pour modéliser les buts, les objectifs et leurs dépendances ;
- SA : « *Structured Analysis* » est une méthode permettant d'analyser les exigences en les convertissant en spécifications exprimées par des modèles ;
- BPMN : « *Business Process Modeling Notation* » est un standard de modélisation graphique des processus métier.

D'autres standards connus ne seront pas repris dans ce travail comme :

- IDEF : « *Integrated Definition Methods* » ;
- SysML : « *Systems Modeling Language* ».

2.1. Causes (Rationale)

Description

Une traduction du mot « *Rationale* » fut difficilement choisie pour ce travail car nous hésitions t entre deux termes s'équivalant en validité : « raisons » ou « justifications ». En effet, dans le contexte de l'ingénierie des exigences, les causes sont des raisons ou arguments qui justifient les décisions prises pour une exigence[3]. Elles regroupent différents concepts tels que :

- Un but métier ou un besoin techniques à atteindre ;
- Une hypothèse ;
- Une contrainte ;
- Une stratégie ;
- Un contexte ;
- Une partie prenante ;
- Un problème.

[1] Par souci de lisibilité, nous dénommerons le terme « objet de modélisation » par l'abréviation « OM ».
[2] (Seilevel, 2012)
[3] (Alexander, et al., 2009)

La modélisation des causes permet de mettre en relation ces concepts entre eux ou d'établir des liens avec des exigences et éléments de solution.

Les modèles de causes peuvent améliorer la compréhension des stratégies au sein de l'ensemble des parties prenantes. Ils servent également à éviter de répéter des débats qui ont déjà été solutionnés en remontrant facilement l'historique de décision[1].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe I.

- Notation GSN (Goal Structuring Notation)
- Modèle de causes (Rationale Model)
- i* : Strategic Rationale
- KAOS : Modèle de buts (Goal Model)

2.2. Concepts du domaine

Description

Afin de pouvoir comprendre la terminologie utilisée par le business et se concentrer productivement sur les exigences, il est nécessaire de représenter les différents concepts et de se renseigner sur ceux qui sont étrangers[2]. Cette compréhension est nécessaire pour éliciter, comprendre, modéliser, spécifier et faire valider les exigences[3].

En ingénierie des exigences, les modèles de concepts permettent de réaliser une représentation des données et de leurs relations selon la vision du métier, par opposition avec la vision technique. Dans ce cas de figure, il ne s'agit pas de réaliser la modélisation d'une base de données. En effet, un diagramme de données peut être utile pour la compréhension de l'agencement des concepts d'un système sans pour autant que la conception technique de celui-ci ne nécessite une base de données[4]. Les relations et leurs cardinalités constituent des exigences à découvrir chez les parties prenantes du métier[5].

Si l'attention est portée sur les actions qu'un système peut opérer sur les données (comme par exemple créer, lire, mettre à jour, copier, déplacer et supprimer), des exigences peuvent être identifiées directement depuis les modèles de concepts. De ces actions émaneront des exigences sur des processus ou des cas d'utilisation que l'analyste devra éliciter et prioriser[6].

La construction d'un modèle de concepts est généralement effectuée de concert avec la réalisation d'un dictionnaire des données venant le compléter par des définitions des concepts représentés. Le

[1] (Origin Consulting (York) Ltd, 2011)
[2] (Beatty, et al., 2012)
[3] (Podeswa, 2009)
[4] (Wiegers, 2003)
[5] (Beatty, et al., 2012)
[6] (Beatty, et al., 2012)

choix de ne pas utiliser de dictionnaire de données existe également. Dans ce dernier cas, et si le modèle adopté le permet, l'analyste indiquera dans le schéma les différents attributs qui qualifient chacun des concepts[1].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe II.

- UML : Diagramme de classes (Class Diagram)
- Schéma Entités-Associations (Entity-Relationship Diagram)
- RML : Diagramme de données métier (Business Data Diagram)

2.3. Découpage du projet

Description

Les techniques de découpage de projet sont rassemblées dans une tâche connue également sous l'anglicisme « *packaging* » d'exigences. Ce terme fait référence à l'action de rassembler les exigences en différents groupes, et cela de façon à pouvoir les communiquer, les comprendre, les valider et les utiliser[2].

Les « paquets » d'exigences peuvent être présentés de plusieurs manières, le format est à choisir en fonction du projet et des parties prenantes à qui il s'adresse. Les critères influents à ce niveau sont la clarté, la précision et le niveau de détail[3] dont par exemple, la présence ou non de documentation ou les dépendances.

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe III.

- UML : Diagramme de Package (Package Diagram)
- Structure de découpage du projet (Work Breakdown Structure)
- RML : Arbre des fonctions (Feature Tree)

2.4. Enchaînement des états

Description

L'ingénierie des exigences peut tirer parti de la modélisation des états de certains objets-clé pour la représentation de contraintes les concernant et la compréhension de leur cycle de vie[4].

[1] (Alexander, et al., 2009)
[2] (International Institute of Business Analysis, 2009)
[3] (International Institute of Business Analysis, 2009)
[4] (Podeswa, 2009)

Il est parfois difficile de se représenter certains objets impliqués dans des processus complexes, de telle façon que malgré la compréhension de ceux-ci, il est difficile de savoir quels sont les états qu'il rencontre et dans quel ordre les transitions peuvent survenir. S'attarder sur la modélisation du cycle de vie de l'objet offre à l'avance l'assurance que tous les états qu'il rencontre sont connus, ainsi que tous les enchaînements possibles entre ces différents états[1].

Ainsi, toutes les exigences concernant un objet et son évolution sont découvertes, avec en prime l'avantage d'identifier les manquements dans d'autres modèles.

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe IV.

- UML : Diagramme d'états-transitions (State-Machine Diagram)
- RML : Diagramme d'état (State Diagram)

2.5. Flux

Description

La possibilité de modélisation des flux est une fonctionnalité importante qui offre une vue sur l'échange d'information au sein du système. L'information échangée peut être des messages ou des données métier, et la modélisation des flux indique comment cette information se déplace dans le système et y subit des transformations[2].

Le système, quant à lui, est représenté dans les modèles de flux par un assemblage de différents processus (cas d'utilisation ou processus système) et d'entités externes. Cette notation contribue à mieux suivre l'évolution des objets métier au sein du système et évaluer la manière dont les processus partagent l'information. En revanche, ces diagrammes ne représentent pas d'ordre dans l'échange des messages, de condition d'embranchement, ni d'indication sur l'état de ceux-ci[3].

Les flux peuvent être modélisés à différents niveaux d'abstraction. Cependant, lorsqu'est modélisé le niveau sommet (système métier complet), on parlera de modélisation du contexte ou de la portée du produit[4] (voir point 2.8).

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe V.

- SA : Diagramme des flux de données (Data Flow Diagram)
- UML : Diagramme de communication (Communication Diagram)

[1] (Beatty, et al., 2012)
[2] (Podeswa, 2009)
[3] (Beatty, et al., 2012)
[4] (Podeswa, 2009)

2.6. Objectifs du système

Description

La modélisation des objectifs permet de capturer et représenter, à différents niveaux de précision et d'affinement, les différents besoins ou cas d'utilisation d'un système. Le fait d'orienter l'analyse des exigences autour des objectifs facilite l'élicitation, l'analyse, la modélisation et la validation des exigences[1]. Elle présente également l'intérêt de vérifier la complétude et la pertinence des exigences, de préparer la détection de conflits entre elles ou encore de fournir un certain niveau d'abstraction afin de valider des choix et proposer des alternatives[2].

Les buts peuvent être assimilés à des exigences *utilisateur*, souvent fonctionnelles, et de haut niveau (c'est-à-dire peu détaillées)[3].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe VI.

- UML : Diagramme des cas d'utilisation (Use cases Diagram)
- i* : Strategic Dependency
- RML : Modèle d'objectifs métier (Business Objectives Model)

2.7. Parties prenantes

Description

Il relève d'une importance capitale de comprendre que les parties prenantes jouent un rôle crucial dans l'évolution d'un projet. Une partie à forte influence favorise ou défavorise facilement la réussite du projet selon la preuve qu'elle témoignera plutôt d'un comportement de soutien ou de contradiction. Certaines parties sont même cachées et peuvent faire preuve d'une grande influence, c'est pourquoi il est d'autant plus que nécessaire de les identifier et de les analyser[4]. Prenons le cas illustratif d'un directeur ayant investi dans une suite logicielle A et dont le fils se trouve nommé chef de projet d'un logiciel B complémentaire au cours duquel l'étude de l'interopérabilité avec A est étudiée. Ce parent est bien entendu dissimulé, néanmoins son influence demeure active auprès du chef de projet qui peut être tenté de pousser l'étude de faisabilité dans un certain sens.

L'analyse des parties prenantes, en outre, permet l'identification de leurs intérêts, des problèmes qui viendraient déranger le projet, des personnes qu'il faut absolument interroger, de celles tenues d'être présentes à chaque réunion, de celles à maintenir informées, des mécanismes d'influences

[1] (van Lamsweerde, 2001)
[2] (van Lamsweerde, 2001)
[3] (Podeswa, 2009)
[4] (Bourne, et al., 2006)

entre les parties prenantes, et enfin des moyens de réduire l'impact des individus à influence négative[1].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe VII.

- Modèle en oignon (Onion Model)
- Matrice d'influence (Influence Matrix)
- RML : Diagramme de l'organisation (Organization Chart)

2.8. Portée du produit

Description

Pour comprendre le produit étudié, il est nécessaire d'identifier les éléments auxquels il est relié dans le monde qui l'entoure[2]. Parmi tous les objets et tâches en relation directe ou indirecte avec le système existant, uniquement certains d'entre eux doivent être effectivement intégrés au produit.

En plus du monde extérieur avec lequel le produit interagit, il y a donc également une portée à définir pour le produit, c'est-à-dire une limite ou une frontière à établir entre les éléments qui feront partie du produit étudié et construit, et ceux qui ne feront partie que de son contexte. Ce dernier comprend les éléments non inclus dans le produit mais avec lesquels le système aura des interactions[3].

Les exigences ne peuvent être élicitées et spécifiées correctement et complètement, qu'à la condition que tous les aspects (personnes, systèmes existants, processus, événements, documents) en relation avec le système soient identifiés[4]. Ainsi, il relève d'un rôle supplémentaire de l'analyse et de la modélisation du produit que d'identifier ces éléments.

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe VIII.

- SA : Diagramme de contexte (Context Diagram)
- Graphe de décomposition fonctionnelle (Functional Decomposition Chart)
- RML : Carte de l'écosystème (Ecosystem Map)

[1] (Babou, 2008)
[2] (Robertson, et al., 2006)
[3] (Alexander, et al., 2009)
[4] (Pohl, et al., 2011)

2.9. Priorité des exigences

Description

Le but de cette fonctionnalité est d'identifier la ou les exigences les plus essentielles dans la mise en place d'un système d'informations.

En effet, le développement d'un tel système est souvent lié à des contraintes de coût, de ressources ou de temps, obligeant les analystes, avec l'aide de l'entièreté des parties prenantes, à faire une sélection d'exigences parmi toutes celles qui ont été élicitées. Une fois les priorités des exigences établies, celles-ci pourront servir de base à la détermination de leur ordre d'implémentation, ou plus précisément, dans quel release il appartiendra d'implémenter quelles exigences[1].

Une des clés de ces analyses de priorité est donc de résoudre le problème en respectant au maximum les contraintes précédemment citées, tout en conservant les perceptions et les attentes d'un maximum de parties prenantes en donnant une place préférentielle aux exigences les plus cruciales pour le projet. Cette difficulté est augmentée lorsque les parties prenantes sont éloignées par la situation géographique, culturelle ou séparées par des frontières de langue et de fuseau-horaire[2].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe IX.

- Processus d'analyse hiérarchique (*Analytic Hierarchy Process*)
- Matrice QFD (*Quality Function Deployment)*
- Calcul du retour sur investissement (*Return On Investment*)

2.10. Processus métier

Description

Un processus métier décrit la réalisation d'une tâche de l'entreprise en la découpant en sous-tâches ou en actions (selon le niveau détail), et en attribuant chacune d'elles aux acteurs les effectuant[3].

ITIL® définit un processus métier de la manière suivante[4] :

> Processus dont la propriété et l'exécution reviennent au business. Un processus business contribue à la fourniture d'un produit ou d'un service à un client du business. Par exemple, un revendeur peut avoir un processus d'achat qui contribue à fournir des services à ses clients business (sa clientèle). De nombreux processus business dépendent des services informatiques.

[1] (Wiegers, 2003)
[2] (Ahmad, et al., 2010)
[3] (Podeswa, 2009)
[4] (Cabinet Office, 2011)

La modélisation de ces processus représente visuellement le séquencement des activités (tâches ou actions), leurs conditions d'occurrence et les acteurs impliqués dans la réalisation de chacune des tâches. Cette modélisation devient indispensable lorsque les processus sont complexes et arborent beaucoup d'embranchements.[1]

Ils peuvent être utilisés pour représenter le processus tel qu'il se trouve (« *as-is* »), ou tel qu'il est attendu (« *to-be* »)[2].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe X.

- UML : Diagramme d'activité (Activity Diagram)
- BPMN : Diagramme de processus métier (Business Process Diagram)
- RML : Flux de processus (Process Flow)

2.11. Scénarios d'utilisation

Description

Les scénarios d'utilisation décrivent le comportement du système face à l'utilisateur. Habituellement, un scénario détaille un cas d'utilisation en décrivant les réponses du système en fonction des inputs du client, et énonçant les différents cas de figure possibles.

Certains modèles offrent uniquement une vision globale en ne présentant que l'enchaînement des écrans du système.

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe XI.

- Prototypage
- « Story-board »
- UML : Diagramme d'activité (Activity Diagram)
- RML : Enchaînement des interfaces utilisateur (UI Flow)

2.12. Scénarios système

Description

Les scénarios système illustrent la manière dont le système se doit de réagir dans un cas d'utilisation précis ainsi que les interactions survenant entre ses différents éléments (acteurs, objets métier, composants).

[1] (Beatty, et al., 2012)
[2] (Podeswa, 2009)

Ils offrent une notion de séquence utile pour détailler des scénarios d'utilisation (voir point 2.11), chaque scénario (normal ou alternatif) faisant l'objet d'un diagramme séparé. La différence entre les scénarios système et les scénarios d'utilisation réside dans le fait que ces derniers ne proposent pas de vue sur ce qu'il se passe « à l'intérieur » du système. Les scénarios système offrent par cette information supplémentaire la possibilité de rentrer dans un niveau plus profond d'abstraction[1].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe XII.

- UML : Diagramme de séquence (Sequence Diagram)
- RML : Flux de système (System Flow)

2.13. Traçabilité des exigences

Description

La traçabilité des exigences est capitale en IE car elle intervient dans de nombreuses phases du projet. Avant-arrière depuis les buts en passant par les différents niveaux d'exigences jusqu'aux tests, ou gauche-droite entre les exigences de même niveau, les liens de traçabilité permettent de maintenir une certaine consistance entre le produit et tout son contexte[2].

Wiegers décrit les différents liens de traçabilité dans son ouvrage *Software Requirements (2nd Edition)*[3], et les rassemble dans un schéma que nous pouvons retrouver dans la Figure III-2. Nous pouvons remarquer que les différentes phases de l'IE sont représentées au travers des éléments qu'il est effectivement possible de relier aux exigences.

[1] (Podeswa, 2009)
[2] (Robertson, et al., 2006)
[3] (Wiegers, 2003)

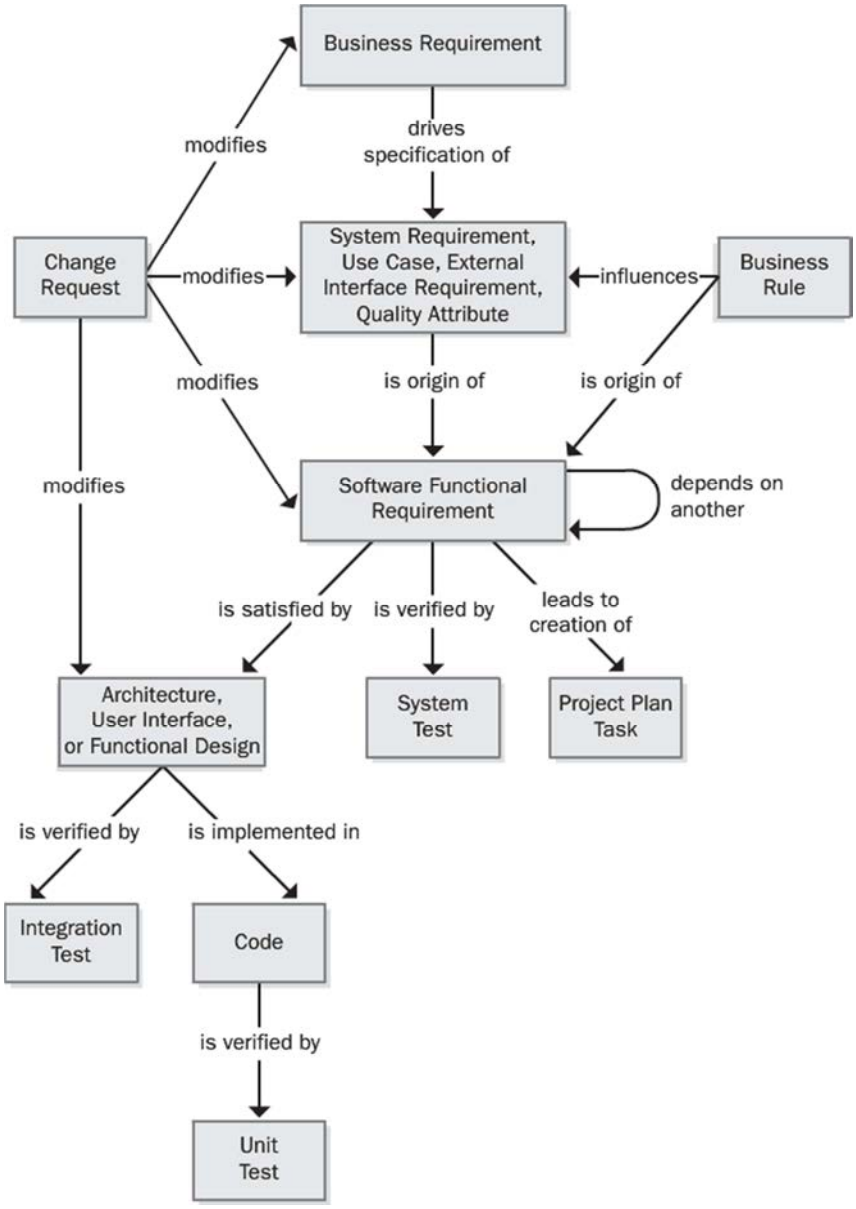

Figure III-2 - Les différents liens de traçabilité des exigences selon K.E. Wiegers, 2003

Une remarque peut être annotée sur ce schéma, les liens y sont représentés fléchés mais la traçabilité est bidirectionnelle : elle permet d'identifier tous les éléments du projet qui dépendent d'une exigence, et inversement, depuis chaque élément du projet, elle permet d'identifier la ou les exigences qui ont conduit à son existence[1].

Modèles et standards courants

Les aperçus des modèles se trouvent en Annexe XIII.

- Table de traçabilité
- Ensemble de matrices de traçabilité

[1] (Robertson, et al., 2006)

3. Résumé des objets de modélisation

Par ce dernier point du chapitre, nous avons rassemblé les connaissances précédemment exposées sous forme de « tableau résumé » reprenant successivement chaque OM. Par souci de clarté, ils sont repris et synthétisés par une question à laquelle un de leurs standards apporterait une réponse.

Tableau III-A - Résumé des OM

Causes (*Rationale*)	Quelles sont les raisons sous-jacentes des objectifs et des décisions ?
Concepts du domaine	Quels sont les objets métier que le système devra traiter ?
Découpage du projet	Comment le système devra-t-il être structuré ?
Enchaînement des états	Comment seront modifiés les objets métier au sein du système ?
Flux	Quels seront les échanges entre le système, ses composants et son contexte ?
Objectifs du système	Qu'est-ce que le système doit accomplir ?
Parties prenantes	Quels sont les intervenants pour la réalisation du projet et de la solution ?
Portée du produit	Quelles sont les limites décrivant la solution ?
Priorité des exigences	Quelles sont les exigences les plus importantes aux yeux des parties prenantes ?
Processus métier	Quel est le déroulement des processus que le système devra respecter ?
Scénarios d'utilisation	Comment le système va répondre à un utilisateur ?
Scénarios système	Comment le système va réaliser un objectif ?
Traçabilité des exigences	Quelles sont les dépendances amont/aval et gauche/droite d'une exigence ?

CHAPITRE IV.

LES MODÈLES ET LEURS APPORTS

1. Introduction

Dans le chapitre précédent, nous avons exposé les arguments en faveur de l'ajout de modèles dans la spécification des exigences. Les différents objets modélisables ont été définis selon leur capacité à mettre en exergue un ou plusieurs aspects des exigences. Enfin, pour chaque type de modèle, nous avons cité des standards existants aptes à fournir une représentation adéquate de l'information.

Nous rappelons que les détails, avantages, et inconvénients de ces standards peuvent être consultés en annexes dans le 0

Dans le présent chapitre, nous aborderons les différents apports que peut avoir la modélisation des exigences, en développant une typologie des modélisés, et les standards de modélisation.

Différentes perspectives de l'analyse des exigences

Dans un premier temps, nous comparerons les différents OM selon qu'ils englobent une ou plusieurs perspectives dans la définition des exigences.

Chacun d'eux apporte une ou plusieurs perspectives sur la définition des exigences[1]. Dans ce point, nous allons classifier et expliquer les perspectives de l'analyse des exigences que peuvent apporter les différents modèles : les objectifs, le comportement, le fonctionnement, les données, et les personnes.

Nous montrerons que pour cibler une perspective particulière, il y a généralement un type de modèle qui est plus adapté. Parallèlement, nous montrerons que chaque modèle est généralement conçu pour représenter une perspective définie, mais sans pour autant s'y limiter.

Différents degrés de fonctionnalité

Globalement, les avis d'experts sont partagés sur la question de l'opportunité d'user ou non des modèles dans la spécification des exigences. D'aucuns jugent que les modèles sont peu adaptés [2], tandis que d'autres admettent qu'ils peuvent convenir avec toutefois, la limite de n'offrir que la capacité à présenter des exigences fonctionnelles.

Nous verrons dans un deuxième temps quels standards peuvent représenter les exigences fonctionnelles et lesquels permettent d'inclure des exigences non-fonctionnelles.

Différents niveaux d'abstraction des exigences

Enfin, dans un troisième temps, nous verrons pour chaque standard, quels différents niveaux d'abstraction des exigences il peut représenter. Nous rappelons que les définitions des niveaux d'abstraction s'établissent comme suit : exigences métier, exigences utilisateur, exigences système. Au fur et à mesure de l'approfondissement et la subdivision du modèle, chaque niveau peut comporter différents degrés de détail au fur et à mesure que l'on approfondit et subdivise le modèle.

[1] (Pohl, et al., 2011)
[2] (Alexander, et al., 2009)

2. Perspective présentée

A la suite de l'étude du chapitre IV, nous avons répertorié un total de cinq perspectives que les modèles peuvent apporter pour la définition des exigences.

Ces perspectives sont les objectifs, le comportement, le fonctionnement, les personnes et enfin les données. Nous représentons leurs imbrications dans la Figure IV-1.

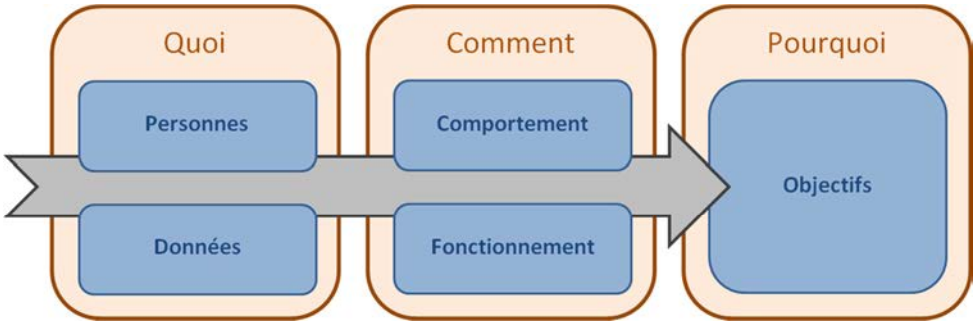

Figure IV-1 - Cinq perspectives des exigences

Des modèles de chacune des perspectives sont indispensables lorsqu'il s'agit d'analyser une solution informatique. En effet, le fait d'analyser le système selon ces différents points de vue, permet de s'assurer que la solution a été examinée sous tous ses angles[1].

Les perspectives seront décrites au fur et à mesure du chapitre. Pour la mise au point de cette catégorisation, nous nous sommes inspirés de plusieurs classifications existantes des perspectives apportées par les modèles.

Classification de l'OMG

En premier, celle de l'*Object Management Group*, qui classifie dans la définition du standard UML ses modèles en trois groupes répartis comme suit[2] :

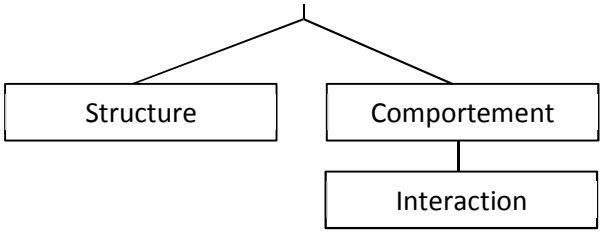

Figure IV-2 - Les grands groupes de modèles en UML

- **Structure** : Les concepts statiques présents dans ou autour du système modélisé.
- **Comportement** : Le déroulement des actions effectuées par le système.
- **Interaction** : Les échanges opérés au sein du système.

[1] (Beatty, et al., 2012)
[2] (Object Management Group, Inc, 2012)

Cette classification est intéressante par la présence d'une distinction entre les diagrammes de comportement et ceux d'interaction. Cependant, de par la vocation originelle d'UML, elle ne représente pas vraiment une idée précise de la perspective des exigences.

Classification de Pohl et Rupp

Dans leur ouvrage *Requirements Engineering Fundamentals*, les auteurs *Pohl et Rupp* classent les modèles dans trois perspectives qui ressemblent à celles de l'OMG mais qui ont été adaptées pour la définition des exigences.

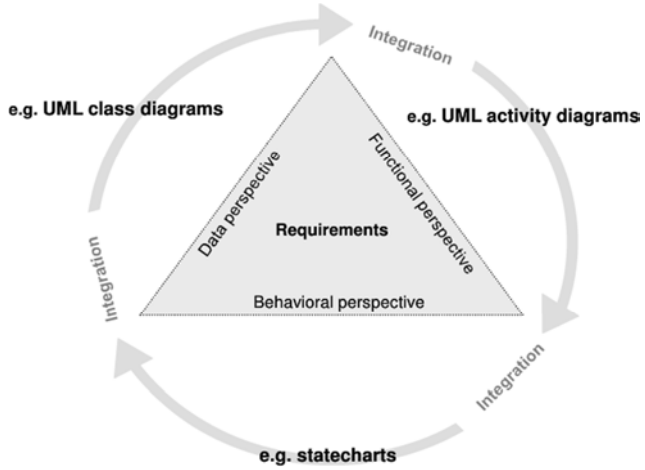

Figure IV-3 - Trois perspectives de représentation des exigences (Pohl, et al., 2011)

Les trois perspectives qu'ils définissent et qui sont représentées dans le schéma de la Figure IV-3 sont les suivantes[1] :

- **Données** : Modélisation des données d'entrées/sorties du système et de leurs interdépendances.
- **Fonctionnement** : Modélisation de la manipulation de l'information au sein du système et entre le celui-ci et son contexte.
- **Comportement** : Modélisation de la réactivité du système face aux événements, des conditions de déclenchement, et des effets du système sur son environnement.

Cette division des perspectives reste valable et attrayante, avec néanmoins la faiblesse d'omettre fortement une notion d'objectifs (exigences de haut niveau que le système doit satisfaire). Cette affirmation est d'autant renforcée par le fait que l'auteur explique les diagrammes de cas d'utilisation UML avant de définir son modèle de perspectives.

Classification de Beatty et Chen

Les auteurs du langage de modélisation RML, dédié spécifiquement aux exigences, dessinent également une classification des différentes perspectives qu'il est possible d'avoir sur elles. Leur catégorisation est très complète, et est illustrée intelligemment au moyen d'une forme triangulaire

[1] (Pohl, et al., 2011)

comme le montre la Figure IV-4. En effet, à la base du triangle sont placés les fondements du système et au sommet est érigée la finalité de celui-ci.

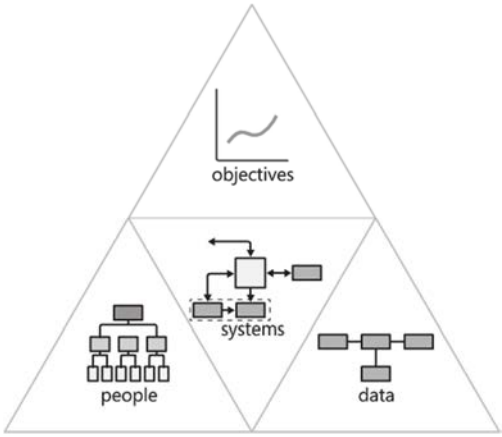

Figure IV-4 - Classification des modèles RML (Beatty, et al., 2012)

Leur classification se fait en quatre perspectives définies comme suit[1] :

- **Objectifs** : Décrit la valeur ajoutée du système au métier.
- **Personnes** : Décrit qui utilise le système.
- **Données** : Décrit les relations entre les objets métier et leur cycle de vie.
- **Systèmes** : Décrit les systèmes existants, les systèmes attendus et comment ils se comportent et interagissent.

Cette répartition des perspectives se rapproche de ce que l'on a observé de l'apport des modèles, mais nous avons jugé opportun d'apporter une distinction supplémentaire. En effet, sachant que le comportement du système et son fonctionnement sont deux perspectives différentes ayant chacune leurs spécificités et leurs modèles propres, il est intéressant de les distinguer.

2.1. Objectifs

La perspective des objectifs décrit les besoins des parties prenantes et leurs attentes vis-à-vis de l'utilisation de celui-ci. Cette dernière doit apporter une valeur au business et aux utilisateurs en solutionnant les problèmes du métier[2].

Cette perspective définit également les limites ou frontières de la solution afin que le produit ne comprenne pas de fonctionnalités superflues et puisse comporter uniquement que ce qui correspond aux objectifs centraux du métier.

Enfin, les objectifs apportent une vue sur les exigences prioritaires pour les parties prenantes et ciblent celles qui répondent à leurs besoins premiers.

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

2.1.1. OM central de la perspective

L'OM central d'une perspective est celui auquel l'analyste pense nécessairement en premier lieu lorsqu'il désire modéliser les exigences selon cette perspective.

Objectifs du système

La modélisation des objectifs du système est manifestement la meilleure manière d'obtenir une représentation globale sur les objectifs du produit. Ces modèles expriment les attentes fonctionnelles du système et sont orientés sur les besoins des utilisateurs[1].

2.1.2. OM dont la perspective est centrale

Certains OM ont pour principale vocation d'exprimer des exigences selon une perspective définie, ce sont ceux-ci que nous désignons dans cette catégorie.

Causes (Rationale)

Les dépendances des problèmes à résoudre par le système peuvent être exprimées dans les modèles de causes. Ces problèmes obtiennent pour réponse des objectifs et des besoins métier qui sont représentés dans les standards de cet OM[2].

Découpage du projet

La modélisation du découpage du organise visuellement les exigences utilisateurs autour de leurs objectifs majeurs. Ces derniers sont affinés progressivement au fur et à mesure des divisions opérées, pour arriver finalement à certaines exigences fonctionnelles.

Le découpage du projet élabore également des définitions de packages qui pourront être priorisés afin de répondre plus adéquatement aux objectifs du métier[3].

Portée du produit

Si le produit est tenu de répondre à des objectifs, il y a également des systèmes en place qu'il doit s'abstenir d'implémenter, mais avec lesquels il devra interagir. La modélisation de la portée du produit offre cette perspective de « frontière logicielle » à respecter[4].

Priorité des exigences

Demander aux parties prenantes d'exprimer un degré de priorité aux exigences procure une mise en avant de leurs objectifs les plus importants, sinon ceux qui les satisferont le plus[5].

[1] (Podeswa, 2009)
[2] (Alexander, et al., 2009)
[3] (International Institute of Business Analysis, 2009)
[4] (Beatty, et al., 2012)
[5] (Robertson, et al., 2006)

Traçabilité des exigences

Toute l'information de l'IE est mappée dans les modèles de traçabilité des exigences. Ils offrent une perspective de dépendance entre les objectifs et les exigences qui en découlent, tout en permettant de traverser les différents niveaux d'abstraction et les différents degrés de détail[1].

2.1.3. Autres OM offrant cette perspective

D'autres OM peuvent être étudiés pour obtenir un vue des exigences selon cette perspective, sans pour autant qu'elle ne soit la fonction principale de ceux-ci.

Processus métier

La modélisation des processus métier n'offre pas une vue sur l'entièreté des objectifs du produit, mais en revanche, elle apporte un degré de précision supplémentaire sur la manière dont chaque objectif métier devra être réalisé[2].

Scénarios système

Parallèlement, les modèles de scénarios système reprennent, par cas d'utilisation, le déroulement des étapes de réalisation d'un objectif des futurs utilisateurs du produit[3].

2.2. Comportement

La perspective du comportement du système est donnée par la modélisation de la succession des situations dans lesquelles le système se trouve. L'évolution de ces situations est idéalement représentée par le biais d'automates finis[4]. Ils mettent en lumière la réactivité du système dans son environnement, ou à l'inverse, l'effet que le système provoque sur celui-ci. Les automates de comportement décrivent également les événements déclencheurs ayant un impact sur le système et la nature de cet impact[5].

2.2.1. OM central de la perspective

Enchaînement des états

Les modèles d'enchaînement des états constituent la meilleure réponse au besoin d'une perspective du comportement du système. Ils donnent une vue sur un objet métier en détaillant son cycle de vie. La vision proposée possède des qualités de concision, d'exhaustivité et de consistance en faisant l'état du système à un instant précis[6].

[1] (Robertson, et al., 2006)
[2] (Beatty, et al., 2012)
[3] (Pohl, et al., 2011)
[4] (Wiegers, 2003)
[5] (Pohl, et al., 2011)
[6] (Wiegers, 2003)

2.2.2. OM dont la perspective est centrale

Flux

La modélisation des flux vient compléter l'enchaînement des états dans la perspective du comportement du produit. En effet, elle rend manifeste la manière dont sont opérés les déplacements et les modifications des objets métier au travers des processus du système[1].

2.2.3. Autres OM offrant cette perspective

Portée du produit

La modélisation de la portée du produit met en avant les événements extérieurs auxquels le système doit réagir. Peuvent également être représentés des composants qui permettront de piloter le système comme par exemple des organes de contrôle[2].

Scénario d'utilisation

Ces modèles décrivent le comportement du système face aux utilisateurs et affichent les réponses qu'il doit apporter face à leurs entrées.

2.3. Fonctionnement

La perspective du fonctionnement du système permet de décrire comment le système traite l'information et quelles transformations il opère sur les données d'entrée pour arriver aux données de sortie[3]. Une vue sur les exigences en termes de déroulement du traitement de ces informations y est également donnée en y décrivant les scénarios à prévoir et les éléments décisionnels pour la réalisation de chacun d'eux[4].

2.3.1. OM central de la perspective

Scénarios système

Les scénarios système offrent la vue la plus détaillée sur le fonctionnement interne du système en incluant les interactions des différents composants de celui-ci. Une importante notion d'ordre vient assurer la complétude de la description du fonctionnement du système.

[1] (Podeswa, 2009)
[2] (Robertson, et al., 2006)
[3] (Pohl, et al., 2011)
[4] (Podeswa, 2009)

2.3.2. OM dont la perspective est centrale

Scénarios d'utilisation

Les scénarios d'utilisation définissent les déroulements d'un cas d'utilisation, expriment les exigences pour un cas normal et celles pour les cas alternatifs. Il s'agit de modéliser le fonctionnement global du système en fonction des besoins des utilisateurs[1].

2.3.3. Autres OM offrant cette perspective

Flux

La modélisation des flux offre une perspective sur la manière dont l'information est échangée et transformée au sein du système.

2.4. Données

Les exigences peuvent également être exprimées selon une perspective sur les données du domaine d'application, ou objets métier.

Les vues importantes sur les objets métier traités par le système sont : la représentation de leur nature ou de leur contenu, la définition des actions perpétrées par le système sur ceux-ci, et les relations qui existent entre eux selon la vision des parties prenantes du métier[2].

La modélisation de leur cycle de vie (création, modification, utilisation, déplacement, copie ou suppression) et de leur utilisation dans les éléments décisionnels fait également partie de ce que cette perspective peut apporter[3].

2.4.1. OM central de la perspective

Concepts du domaine

Les modèles de concepts du domaine permettent de définir un ensemble d'objets métier en annotant les attributs qui les caractérisent, et des exigences telles que les actions que le système a sur eux, ou les relations que les utilisateurs voient entre eux, ainsi que la multiplicité de ces relations[4].

2.4.2. OM dont la perspective est centrale

Nous n'avons pas identifié d'autre OM dont les données soient la perspective centrale.

[1] (Wiegers, 2003)
[2] (Beatty, et al., 2012)
[3] (Beatty, et al., 2012)
[4] (Wiegers, 2003)

2.4.3. Autres OM offrant cette perspective

Enchaînement des états

Le cycle de vie d'un objet métier, ou d'une donnée métier en particulier, peut être analysé et modélisé par le biais de cet OM[1]. Ces modèles indiquent également les processus à travers lesquels l'objet métier sera manipulé.[2]

Flux

Les données échangées au sein du système sont présentées par les modèles de flux. Ces derniers permettent d'obtenir un aperçu et une compréhension de l'éventail des données au sein du domaine d'activité[3].

2.5. Personnes

La perspective des personnes doit donner en premier un aperçu sur les parties prenantes du projet dont font partie tous les utilisateurs du système.

Ces utilisateurs sont mis en relation avec leurs objectifs, avec les processus qu'ils doivent pouvoir initier dans le système et avec les exigences qu'ils posent sur son utilisation.

2.5.1. OM central de la perspective

Parties prenantes

Les modèles de parties prenantes décrivent toutes les personnes qui ont un rapport proche ou lointain avec le projet. Certain modèles permettent également de modéliser une hiérarchie au sein de celles-ci ou de représenter leur fonction dans le contexte du projet[4].

2.5.2. OM dont la perspective est centrale

Processus métier

La modélisation des processus métier énumère les tâches effectuées par les personnes et détaillent les étapes à réaliser ainsi que les décisions prises par les utilisateurs pour atteindre l'effet escompté[5].

[1] (Podeswa, 2009)
[2] (Wiegers, 2003)
[3] (International Institute of Business Analysis, 2009)
[4] (Robertson, et al., 2006)
[5] (Beatty, et al., 2012)

2.5.3. Autres OM offrant cette perspective

Causes (Rationale)

Les causes d'existence de certaines exigences peuvent être des personnes ou des contraintes exprimées spécifiquement par des parties prenantes et dont le lien est intéressant à modéliser[1].

Objectifs du système

Les modèles d'objectifs du système relient les utilisateurs aux fonctionnalités qu'ils exigent du système[2].

Scénarios d'utilisation

Les modèles de scénarios du système expriment les interactions entre le système et les utilisateurs ou d'éventuels autres acteurs gravitant autour de celui-ci[3].

Traçabilité des exigences

Tous les liens entre les exigences peuvent apparaître dans les modèles de traçabilité de sorte à pouvoir identifier celles dérivant des besoins des parties prenantes du métier, des besoins utilisateurs, ou encore des contraintes imposées par d'autres parties prenantes[4].

[1] (Alexander, et al., 2009)
[2] (Podeswa, 2009)
[3] (Wiegers, 2003)
[4] (Wiegers, 2003)

2.6. Récapitulatif

Tableau IV-A - Les perspectives présentées par les OM

	Objectifs	Comportement	Fonctionnement	Données	Personnes
Causes (Rationale)	x				x
Concepts du domaine				xx	
Découpage du projet	x				
Enchaînement des états		xx		x	
Flux		x	x	x	
Objectifs du système	xx				x
Parties prenantes					xx
Portée du produit	x	x			
Priorité des exigences	x				
Processus métier	x				x
Scénarios d'utilisation		x	x		x
Scénarios système	x		xx		
Traçabilité des exigences	x				x

Légende :

- xx OM central de la perspective
- x OM dont la perspective est centrale
- x OM offrant cette perspective

3. Degré de fonctionnalité

La plupart des OM sont orientés vers les exigences fonctionnelles. Cependant, certains permettent de représenter ou découvrir les exigences non-fonctionnelles également. D'autres encore, aident l'analyste dans sa tâche d'IE mais n'ont pas la vocation explicite de spécifier les exigences.

Nous avons également constaté que certains standards d'un OM disposent d'une syntaxe plus à même de représenter un degré de fonctionnalité que d'autres standards ne permettent pas.

Dans ce point seront donc stipulés les standards de modélisation qui permettent de spécifier des exigences fonctionnelles ou non fonctionnelles, de les schématiser, ou d'y apporter des informations pertinentes. Certains modèles procurent également une découverte de nouvelles exigences, ils seront mis en avant également. En revanche, ne seront pas catalogués ceux qui ne font que citer une exigence sans la définir ou sans apporter d'information complémentaire.

3.1. Exigences fonctionnelles

Causes (Rationale)

Les causes peuvent être mises en relation avec divers éléments, dont les objectifs (dits « *goals* ») qui sont pour la plupart des expressions d'un besoin dont découlent des exigences fonctionnelles[1]. Les standards aptes à représenter ces exigences, sous la forme de réponse apportée à un objectif sont la Notation GSN, le modèle Strategic Rationale.

Concepts du domaine

Tous les modèles de concepts que nous avons identifiés, à savoir le Diagramme de classes, le Schéma Entités-Associations et le Diagramme de données métier, permettent à l'analyste de déduire et de modéliser certaines exigences fonctionnelles exprimées par les cardinalités des relations entre les concepts[2].

Le Diagramme de classe a un avantage supplémentaire car il permet de représenter, par concept, les activités avec lesquelles ce concept peut interagir.

Découpage du projet

Un Arbre des fonctions permet de découvrir des exigences en identifiant les fonctionnalités manquantes de certaines fonctions, ou en le mettant en comparaison avec d'autres modèles[3].

Les autres modèles n'apportent pas réellement d'information supplémentaire pertinente en termes d'exigences fonctionnelles.

[1] (Alexander, et al., 2009)
[2] (Grunske, et al., 2010)
[3] (Beatty, et al., 2012)

Enchaînement des états

Les modèles d'enchaînements des états permettent de modéliser des exigences fonctionnelles par rapport aux états des objets métier, telles que des modifications à leur apporter en réponse à un évènement[1]. Le Diagramme d'états-transitions d'UML et le Diagramme d'état de RML offrent tous deux cette possibilité. Le modèle d'UML offre un degré supplémentaire en permettant la modélisation des activités qui doivent être exécutées dans des conditions précises.

Enfin, des exigences fonctionnelles peuvent être identifiées lorsque sont posées des questions par rapport aux données, comme par exemple « Qu'est-ce qui initie ce changement d'état ? »[2].

Flux

Des exigences fonctionnelles, telles que la nature des données nécessaires à l'exécution d'un processus, ou l'ensemble des processus requis pour la transformation de ces données, sont identifiables et modélisables dans un Diagramme de Flux de données[3]. Le Diagramme de communication ajoute une en plus notion d'ordre.

Objectifs du système

Les grandes fonctionnalités ou tâches des utilisateurs peuvent être modélisées dans un Diagramme des cas d'utilisation. Ce dernier est également un moyen d'identifier quels acteurs peuvent initier ces tâches[4]. Un Modèle d'objectifs métier permet d'indiquer quel objectif une fonctionnalité satisfait et de déduire les manquements[5]. Le Strategic Dependancy offre les possibilités de ces deux modèles.

Portée du produit

Le Diagramme de contexte et la Carte de l'écosystème permettent d'identifier et de représenter les interactions que doit avoir le produit avec les systèmes externes.

Le Graphe de décomposition est une manière graphique d'agencer les activités du système, mais n'offre pas de réelle possibilité de modélisation des exigences fonctionnelles, ni de moyen de déceler des éléments manquants[6].

Processus métier

Les exigences métier à remplir par le système sont décrites sous la forme d'automates finis dans le Diagramme d'activité, le Diagramme de processus métier et le Flux de processus. Ces modèles ont

[1] (Pohl, et al., 2011)
[2] (Beatty, et al., 2012)
[3] (Podeswa, 2009)
[4] (Podeswa, 2009)
[5] (Beatty, et al., 2012)
[6] (International Institute of Business Analysis, 2009)

également tous, avec l'aide des parties prenantes, la capacité de révéler les manquements dans les processus[1].

Scénarios d'utilisation

Des modèles de scénarios d'utilisation tels que le Diagramme d'activité, le Prototypage et l'Enchaînement des interfaces utilisateur peuvent exprimer les exigences fonctionnelles propres à l'interaction entre le système et ses utilisateurs. En les confrontant à ces derniers, ils permettront d'identifier des exigences manquantes ou mal comprises.

Scénarios système

Les scénarios systèmes donnent une modélisation des exigences au niveau de détail le plus élevé, en décrivant tout le fonctionnement des étapes de traitement et les composants qui en dépendent. Aussi, lors des phases de validation, certaines exigences nouvelles peuvent être découvertes grâce à ces deux modèles.

Traçabilité des exigences

Modéliser les liens de traçabilité des exigences dans une Table de traçabilité n'en donne pas une réelle spécification, mais permettent de remonter sous forme graphique aux causes et objectifs qui sont la raison de leur existence.

Les matrices de traçabilité, utilisées par exemple pour lier les exigences au cas d'utilisations, assurent un niveau supplémentaire de complétude.

3.2. Exigences non-fonctionnelles

Causes (Rationale)

La représentation graphique des causes permet de se poser les questions sur les raisons d'existence d'un objectif ou d'une exigence. Ces raisons peuvent être des exigences non-fonctionnelles comme des exigences de portabilité, de fiabilité, de performance ou de coût. Les standards étudiés ont tous cette possibilité : Notation GSN, Modèle de causes, Strategic Rationale et Modèle de buts.

Concepts du domaine

Certaines cardinalités modélisables dans le Diagramme de classes, le Schéma Entités-Associations ou le Diagramme de données métier font référence à des exigences non-fonctionnelles telles que des contraintes légales, de sécurité, ou de performance[2].

[1] (Beatty, et al., 2012)
[2] (Grunske, et al., 2010)

Enchaînement des états

Certains enchaînements d'états sont interdits ou obligatoires en raison de contraintes correspondant à des exigences de sécurité ou de fiabilité. Ces exigences peuvent être indiquées sur les transitions d'un Diagramme d'états-transitions ou d'un Diagramme d'état.

Objectifs du système

Les causes des objectifs modélisables dans un Modèle d'objectifs métier peuvent correspondre à certaines exigences non-fonctionnelles. Le principe de ce modèle étant de partir des problèmes pour arriver aux solutions, ces exigences doivent être connues à l'avance[1].

Un modèle Strategic Dependancy, reliant les objectifs aux acteurs dont ils découlent ainsi qu'à ceux qui en dépendent, offrent un moyen concret de modéliser et dériver des exigences non-fonctionnelles ou de qualité, dénommées « soft-goals » par le standard[2].

Portée du produit

La modélisation de l'inclusion du produit dans son contexte, telle que la rendent le Diagramme de contexte et la Carte de l'écosystème, permettent d'identifier des exigences non fonctionnelles de sécurité, d'interopérabilité, de portabilité ou encore de confidentialité.

Priorité des exigences

Des exigences non-fonctionnelles sont souvent les raisons sous-jacentes de priorités des exigences, qui peuvent être modélisés dans le Processus AHP ou dans une Matrice QFD.

Processus métier

Les modèles de processus métier sont orientés sur le déroulement des processus et n'offrent pas de réel moyen de modéliser des exigences non-fonctionnelles. Cependant certaines exigences de ce type peuvent être déduites si l'analyste se penche sur les raisons du fonctionnement interne d'un processus.

Scénarios d'utilisation

Les modèles de scénarios d'utilisation permettent d'identifier des exigences de fiabilité ou de performance au moyen du Prototypage, d'un Diagramme d'activité et de l'Enchainement des interfaces utilisateur. Des exigences non-fonctionnelles de confidentialité et d'utilisabilité peuvent également y être modélisées.

[1] (Beatty, et al., 2012)
[2] (Alexander, et al., 2009)

Scénarios système

Les modèles de scénario systèmes sont orientés sur le fonctionnement du système et n'offrent pas de réel moyen de modéliser des exigences non-fonctionnelles. Cependant certaines exigences non-fonctionnelles peuvent être déduites si l'analyste examine les raisons du fonctionnement interne d'un processus.

Traçabilité des exigences

Les exigences non-fonctionnelles identifiées peuvent être modélisées dans une Table de traçabilité ou dans une Matrice de traçabilité afin d'exprimer leurs causes, les objectifs qui en dépendent et les exigences fonctionnelles qu'elles ont générées.

3.3. Récapitulatif

Tableau IV-B - Les modèles par rapport aux degrés de fonctionnalité

	Standard	Modèle	Exigences fonctionnelles	Exigences non fonctionnelles
Causes (Rationale)	GSN	Notation GSN (Goal Structuring Notation)	**M**	**DM**
		Modèle de causes (Rationale Model)		**DM**
	i*	Strategic Rationale	**DM**	**DM**
	KAOS	Modèle de buts (Goal Model)		**DM**
Concepts du domaine	UML	Diagramme de classes (Class Diagram)	**DM**	**DM**
		Schéma Entités-Associations (Entity-Relationship Diagram)	**DM**	**DM**
	RML	Diagramme de données métier (Business Data Diagram)	**DM**	**DM**
Découpage du projet	UML	Diagramme de Package (Package Diagram)		
		Structure de découpage du projet (Work Breakdown Structure)		
	RML	Arbre des fonctions (Feature Tree)	**D**	
Enchaînement des états	UML	Diagramme d'états-transitions (State-Machine Diagram)	**DM**	**DM**
	RML	Diagramme d'état (State Diagram)	**DM**	**DM**
Flux	SA	Diagramme des flux de données (Data Flow Diagram)	**DM**	
	UML	Diagramme de communication (Communication Diagram)	**DM**	
Objectifs du système	UML	Diagramme des cas d'utilisation (Use cases Diagram)	**DM**	
	i*	Strategic Dependency	**DM**	**DM**
	RML	Modèle d'objectifs métier (Business Objectives Model)	**D**	**M**
Parties prenantes		Modèle en oignon (Onion Model)		
		Matrice d'influence (Influence Matrix)		
	RML	Diagramme de l'organisation (Organization Chart)		
Portée du produit	SA	Diagramme de contexte (Context Diagram)	**DM**	**D**
		Graphe de décomposition fonctionnelle (Functional Decomposition Chart)		
	RML	Carte de l'écosystème (Ecosystem Map)	**DM**	**D**
Priorité des exigences		Processus d'analyse hiérarchique (Analytic Hierarchy Process)		**D**
		Matrice QFD (Quality Function Deployment)		
		Calcul du retour sur investissement (Return On Investment)		**D**
Processus métier	UML	Diagramme d'activité (Activity Diagram)	**DM**	**D**
	BPMN	Diagramme de processus métier (Business Process Diagram)	**DM**	**D**
	RML	Flux de processus (Process Flow)	**DM**	**D**
Scénarios d'utilisation		Prototypage	**DM**	**D**
	UML	Diagramme d'activité (Activity Diagram)	**DM**	**D**
	RML	Enchaînement des interfaces utilisateur (UI Flow)	**DM**	**D**
Scénarios système	UML	Diagramme de séquence (Sequence Diagram)	**DM**	**DM**
	RML	Flux de système (System Flow)	**DM**	**DM**
Traçabilité des exigences		Table de traçabilité	**M**	**M**
		Ensemble de matrices de traçabilité	**M**	**M**

Légende : **D** Standard permettant de Dériver le degré de fonctionnalité
M Standard permettant de Modéliser le degré de fonctionnalité

4. Niveau d'abstraction

Lorsqu'il s'agit de spécifier les exigences de manière graphique, certains OM conviennent mieux à un niveau d'abstraction tandis que d'autres conviennent à plusieurs niveaux.

Pour chaque niveau d'abstraction, nous examinons dans ce point les OM qui leur sont appropriés.

4.1. Exigences métier

Concepts du domaine

Certains métiers ne disposent que d'un ou deux objets importants tandis que d'autres, d'une panoplie d'objets imbriqués[1]. Dans tous les cas, leur compréhension et leur spécification sont primordiales au bon déroulement de la suite de l'ingénierie des exigences.

Tous les standards étudiés sont adaptés pour la modélisation des données métier, de leurs relations et de règles métier associées aux objets[2]. Chacun des standards assure une connaissance et une cohérence des termes métier utilisés lors de l'analyse[3].

Enchaînement des états

L'étude des états des principaux objets est primordiale pour l'identification de tous les processus métier impliqués dans le cycle de vie de cet objet. En étudiant leur cycle de vie, on peut comprendre et modéliser le comportement des objets d'un point de vue métier[4] au moyen d'un Diagramme d'états-transitions ou d'un Diagramme d'états.

Flux

Le Diagramme de communication est une manière pratique d'analyser et schématiser l'agencement de différents systèmes du métier, les liens existants entre eux et les informations qu'ils échangent[5].

Le Diagramme de Flux de données pourrait être utilisé également à cet escient, mais nous l'assimilons alors à un Diagramme de contexte[6].

Objectifs du système

Les objectifs du système d'un point de vue métier sont également appelés « objectifs métier ». Se concentrer sur cet OM selon une perspective métier est une base importante pour l'étude préalable de la portée du produit ainsi que pour la recherche des éléments qui augmenteront la valeur du produit aux yeux des parties prenantes du métier[7]. Les modèles adaptés sont le Strategic

[1] (Beatty, et al., 2012)
[2] (Podeswa, 2009)
[3] (Robertson, et al., 2006)
[4] (Podeswa, 2009)
[5] (Podeswa, 2009)
[6] (Podeswa, 2009)
[7] (Beatty, et al., 2012)

Dependency pour relier chaque objectif à une partie prenante, et le Modèle d'objectifs métier pour dériver les objectifs depuis les problèmes à résoudre.

Portée du produit

Le Diagramme de contexte et la Carte de l'écosystème représentent un haut niveau d'abstraction sur les interactions du produit avec ses éléments externes[1], qu'ils soient acteurs, terminaux de contrôle, systèmes externes, ou bases de données. Ces interactions constituent des besoins métier tout en définissant la portée du produit, qui a été étudiée pour remplir au mieux les objectifs métier.

Processus métier

Pouvant être utilisé à différents niveaux d'abstraction au moyen d'un Diagramme d'activité, d'un Diagramme de processus métier ou d'un Flux de processus, ces modèles conviennent pour la description des étapes de haut niveau d'un processus[2].

4.2. Exigences utilisateur

Concepts du domaine

Le seul standard étudié qui peut être utilisé pour spécifier des exigences utilisateurs est le Diagramme de classes, qui offre la possibilité d'associer à chaque concept les actions que le système peut opérer sur eux. Ces actions correspondent à des cas d'utilisation et les relations issues de l'objet en question permettront d'identifier les autres concepts qui risquent d'être également impliqués dans le cas d'utilisation[3].

Objectifs du système

Les objectifs du système, assimilés à ce niveau d'abstraction à des cas d'utilisation, sont spécifiés au moyen d'un Diagramme de cas d'utilisation. La sémantique de ce dernier visant des tâches réalisées par un acteur au moyen du système, ce modèle est le moyen le plus efficace pour la découverte et la modélisation des objectifs du système du point de vue des utilisateurs[4].

Le Strategic Dependency, conçu pour la mise en relation de chaque objectif avec une paire de parties prenantes, (l'une dont dépend un objectif et l'autre dépendante de celui-ci), permet d'arriver jusqu'aux besoins des utilisateurs.

Scénarios d'utilisation

Les scénarios d'utilisation sont le moyen le plus propice pour la description des interactions entre un utilisateur et le système. Un prototypage facilitera la découverte de nouvelles exigences utilisateur[5].

[1] (Wiegers, 2009)
[2] (Beatty, et al., 2012)
[3] (Podeswa, 2009)
[4] (Podeswa, 2009)
[5] (Robertson, et al., 2006)

Le Diagramme d'activité, utilisé sans les couloirs, peut décrire les différentes actions que l'utilisateur réalise sur le système et les éléments de décision générant un cas d'utilisation alternatif possiblement représentable dans le même modèle[1]. Enfin, l'Enchaînement des interfaces utilisateur modélise l'application et la manière dont l'utilisateur pourra y naviguer[2].

Scénarios système

Cet OM offre un niveau de détail plus important que celui des scénarios utilisateur. Il est effectivement prévu pour la description des cas d'utilisation et de leurs alternatives, comme c'est le cas du Diagramme de séquence qui fournit une intéressante notion temporelle[3].

Quant au Flux de système, il est conçu pour définir le comportement d'un système autonome. Sa syntaxe est très proche du Flux de processus, donc dès lors qu'il sera utilisé pour le niveau d'abstraction des utilisateurs, il sera assimilé à cet autre standard[4].

4.3. Exigences système

Enchaînement des états

Lors de l'étude d'un système autonome ne comportant aucun objet métier, un Diagramme d'états-transitions ou un Diagramme d'état peut être utilisé pour la modélisation du système complet et des états dans lequel le système lui-même se trouvera, et non plus les objets internes à celui-ci[5]. Même si nous pourrions assimiler dans ce cas le système à un gros objet métier, cet OM présenterait un bon nombre d'exigences propres au système.

Objectifs du système

Le Diagramme des d'utilisation dispose d'une syntaxe pour représenter des acteurs et les actions qu'ils peuvent réaliser dans le système. Bien souvent, les acteurs sont assimilés à des utilisateurs mais ils schématisent toutefois également un système externe. Dans ce cas, le modèle représente également des exigences système[6].

Flux

Le Diagramme de flux de données et le Diagramme de communication décrivent les mouvements de données au sein d'un système. Les processus systèmes sont représentés dans le premier et spécifient quels sont leurs entrées/sorties. Le second montre les messages échangés entre les composants internes qui y sont schématisés.

[1] (Wiegers, 2003)
[2] (Beatty, et al., 2012)
[3] (Podeswa, 2009)
[4] (Beatty, et al., 2012)
[5] (Beatty, et al., 2012)
[6] (Podeswa, 2009)

Portée du produit

Cet OM comporte deux standards qui permettent de modéliser les exigences de communication du système avec les systèmes externes : le Diagramme de contexte et la Carte de l'écosystème.

Scénarios système

Les actions précises des sous-systèmes peuvent être décrites par cet OM, en donnant un vue sur l'information de l'emplacement où ces actions sont produites, autrement dit, le sous-système qui réalise l'action. Le Diagramme de séquence donne des facilités pour schématiser des notions de temporisation et de communication[1], tandis que le Flux de système est plus propice à donner des informations sur les embranchements et conditions avancées des étapes d'exécution d'un système autonome[2].

[1] (Podeswa, 2009)
[2] (Beatty, et al., 2012)

4.4. Récapitulatif

Tableau IV-C - Les niveaux d'abstraction représentés par les modèles

	Standard	Modèle	Exigences métier	Exigences utilisateur	Exigences système
Causes (Rationale)	*GSN*	Notation GSN (Goal Structuring Notation)			
		Modèle de causes (Rationale Model)			
	*i**	Strategic Rationale			
	KAOS	Modèle de buts (Goal Model)			
Concepts du domaine	*UML*	Diagramme de classes (Class Diagram)	x	x	
		Schéma Entités-Associations (Entity-Relationship Diagram)	x		
	RML	Diagramme de données métier (Business Data Diagram)	x		
Découpage du projet	*UML*	Diagramme de Package (Package Diagram)			
		Structure de découpage du projet (Work Breakdown Structure)			
	RML	Arbre des fonctions (Feature Tree)			
Enchaînement des états	*UML*	Diagramme d'états-transitions (State-Machine Diagram)	x		x
	RML	Diagramme d'état (State Diagram)	x		x
Flux	*SA*	Diagramme des flux de données (Data Flow Diagram)			x
	UML	Diagramme de communication (Communication Diagram)	x		x
Objectifs du système	*UML*	Diagramme des cas d'utilisation (Use cases Diagram)		x	x
	*i**	Strategic Dependency	x	x	
	RML	Modèle d'objectifs métier (Business Objectives Model)	x		
Parties prenantes		Modèle en oignon (Onion Model)			
		Matrice d'influence (Influence Matrix)			
	RML	Diagramme de l'organisation (Organization Chart)			
Portée du produit	*SA*	Diagramme de contexte (Context Diagram)	x		x
		Graphe de décomposition fonctionnelle (Functional Decomposition Chart)			
	RML	Carte de l'écosystème (Ecosystem Map)	x		x
Priorité des exigences		Processus d'analyse hiérarchique (Analytic Hierarchy Process)			
		Matrice QFD (Quality Function Deployment)			
		Calcul du retour sur investissement (Return On Investment)			
Processus métier	*UML*	Diagramme d'activité (Activity Diagram)	x		
	BPMN	Diagramme de processus métier (Business Process Diagram)	x		
	RML	Flux de processus (Process Flow)	x		
Scénarios d'utilisation		Prototypage		x	
	UML	Diagramme d'activité (Activity Diagram)	x	x	
	RML	Enchaînement des interfaces utilisateur (UI Flow)		x	
Scénarios système	*UML*	Diagramme de séquence (Sequence Diagram)		x	x
	RML	Flux de système (System Flow)		x	x
Traçabilité des exigences		Table de traçabilité			
		Ensemble de matrices de traçabilité			

Légende :

x OM propice à la modélisation d'exigences à ce degré de fonctionnalité

CHAPITRE V.

LES MODÈLES AU SEIN DU CYCLE DE L'IE

1. Introduction

Dans le chapitre précédent, nous avons vu comment les modèles pouvaient apporter une valeur ajoutée dans le cycle de vie de l'IE.

Nous avons identifié quels étaient les OM plus adaptés à représenter une perspective définie, ainsi que ceux qui la complétaient en y apportant une information supplémentaire.

Nous avons identifié également les standards de modélisation plus adaptés pour découvrir des exigences fonctionnelles ou des exigences non-fonctionnelles.

Enfin, nous avons ciblé les standards de modélisation offrant la possibilité de représenter les exigences à différents niveaux d'abstraction ou, à l'inverse, spécifiques à la modélisation d'un niveau défini.

Nous avons voulu distinguer ce chapitre du précédent car il ne s'agit plus de voir *comment* mais bien *quand* les modèles peuvent apporter une valeur ajoutée dans le cycle de vie de l'IE.

Les modèles sont un allié précieux dans chaque stade ou phase du cycle de vie du développement du projet et plus spécifiquement le cycle de vie de l'IE. Dans ce contexte (par opposition à celui de la conception logique du système), il est important de noter que les modèles ne doivent pas être d'une trop grande précision et beauté, au risque de paraître « illégitimement corrects »[1]. De plus, une modélisation trop minutieuse requiert davantage de temps, au détriment de la compréhension de leur aptitude à assurer la découverte d'un ensemble complet d'exigences[2].

Phase par phase, nous décrirons quels sont les OM pertinent à y utiliser et quelles sont leurs spécificités au sein de chacune d'elles.

Pour terminer nous établirons une mise en comparaison de ces modèles au sein des phases, sous la forme d'un tableau récapitulatif reprenant les différentes phases parcourues dans ce chapitre.

[1] (Alexander, et al., 2009)
[2] (Beatty, et al., 2012)

2. Les phases du cycle de l'IE

2.1. Phase 1 : Élicitation

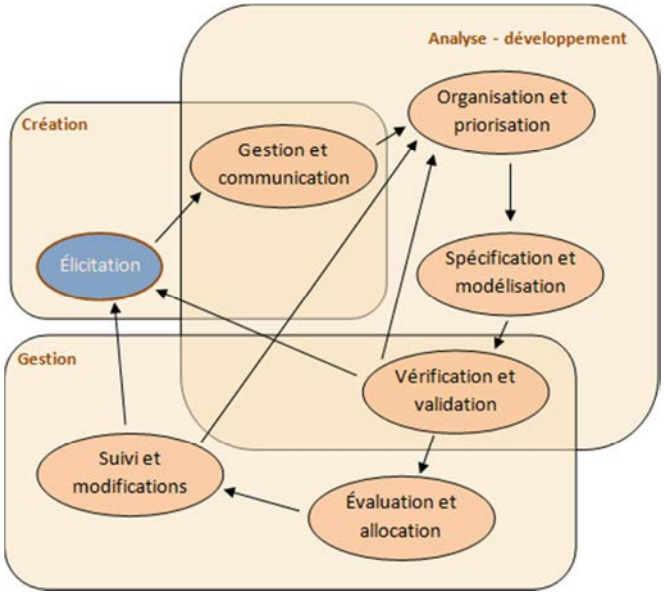

Figure V-1 - L'élicitation dans le cycle de l'IE

2.1.1. Description

Le mot « élicitation » est un anglicisme, utilisé dans la littérature logicielle. Il pourrait être traduit littéralement par « extraction », « obtention », « élucidation », ou expliqué de manière plus adaptée par le terme « découverte ».

L'élicitation est définie effectivement comme un processus de recherche des exigences donnant lieu à une activité essentiellement sociale, de communication et de négociation nécessitant une compréhension du domaine de l'application et une analyse des besoins des parties prenantes.

En effet, c'est lors de cette phase que se réalise l'identification des sources des exigences, que sont entre autres les parties prenantes, en sollicitant une exploration et une compréhension fine des souhaits et besoins des intéressés[1].

En outre, l'élicitation couvre la découverte et la définition des limites du système et l'identification des contraintes imposées pour la solution[2].

[1] (Robertson, et al., 2006)
[2] (Alexander, et al., 2009)

2.1.2. Objets de modélisation

Concepts du domaine

La compréhension et la description des concepts clés du domaine sont facilitées par leur modélisation. La majorité des standards offrent la possibilité d'ajouter des concepts au modèle au fur et à mesure de leur découverte, mais au plus ceux-ci sont définis tôt, au plus la bonne compréhension est assurée au sein de l'équipe technique, et au moins les risques d'ambiguïté peuvent survenir[1].

Enchaînement des états

L'étude du système selon le cycle de vie des objets-clés du domaine permet d'identifier les événements de déclenchement des modifications de l'état de ces objets. Aussi, des exigences relatives aux conditions d'occurrence d'une activité ou d'un processus peuvent être découvertes par le biais de la modélisation des enchaînements des états. Enfin, cette dernière permet l'identification de tous les processus utilisant un objet métier défini.

Objectifs du système

La modélisation des objectifs est un excellent moyen de représenter les exigences des utilisateurs finaux, par la vue de haut niveau d'abstraction qu'ils apportent. Cette vision permet également de maintenir les workshops de démarrage du projet centrés sur les grandes attentes du métier et des utilisateurs du système.

Parties prenantes

Les parties prenantes sont la source principale des exigences. C'est pourquoi il est impératif de les reconnaître rapidement afin de savoir quelle est la personne la plus habilitée à répondre à une question définie. Leur modélisation permet de s'assurer que personne n'a été oublié, de déterminer qui utilisera le système et à quelles personnes il convient de donner des comptes rendus[2].

Portée du produit

La portée du produit doit être analysée rapidement dans le cycle de vie, dès les phases d'élicitation, afin de pouvoir déterminer précisément quel est le domaine d'activité étudié, où se situe le produit dans son contexte, et qu'est-ce qui l'entoure dans son environnement. Il est important, une fois ces éléments définis, de pouvoir fixer la portée du produit afin de ne pas s'attarder sur l'étude de fonctionnalités pour lesquelles le métier n'attend pas d'implémentation[3].

[1] (Podeswa, 2009)
[2] (Robertson, et al., 2006)
[3] (Robertson, et al., 2006)

Processus métier

La compréhension des processus du domaine d'application est facilitée par leur modélisation, qui permet de découvrir les activités à réaliser, leur séquence d'exécution, et les éléments pris en compte dans les décisions nécessaires à obtenir la finalité désirée par le métier[1].

Scénarios d'utilisation

Un premier jet de représentation des scénarios, présenté aux parties prenantes, peut leur ouvrir la voie vers des suggestions sur le fonctionnement qu'ils attendent d'un système.

Des scénarios d'utilisation émanent également certaines règles de précédence telles les préconditions et les conditions de déclenchement[2].

Traçabilité des exigences

Les éléments que l'on aura choisi de tracer doivent être déterminés très tôt dans le projet puisqu'ils devront être enregistrés dès leur existence[3].

[1] (Pohl, et al., 2011)
[2] (Podeswa, 2009)
[3] (Podeswa, 2009)

2.2. Phase 2 : Gestion et communication

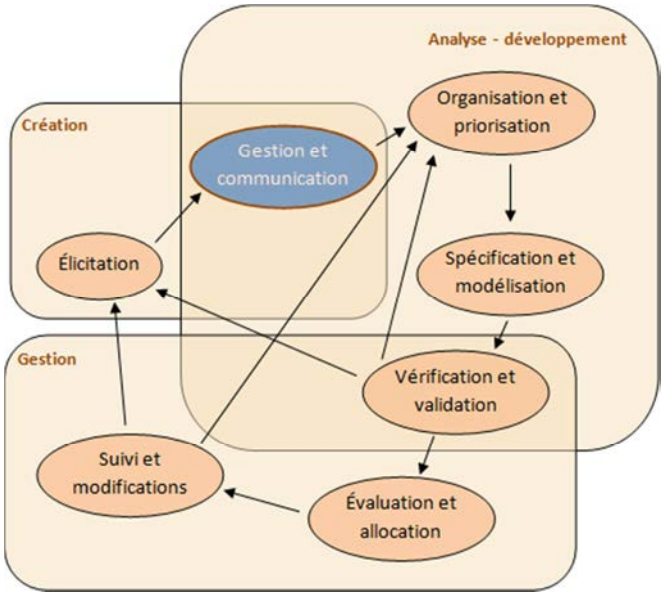

Figure V-2 - La Gestion et la communication dans le cycle de l'IE

2.2.1. Description

La phase de gestion et communication regroupe les différentes tâches d'administration et d'expression des exigences aux diverses parties prenantes, afin de s'assurer qu'elles ont toutes une compréhension claire de la nature de la solution et des exigences qu'elle doit satisfaire.

Les parties prenantes pouvant être issues de différents centres de compétences, il faut pouvoir opter pour un consensus sur l'implémentation des exigences et sur la définition de la portée du projet, en gérant les conflits et inconsistances et facilitant la communication entre les parties prenantes concernées[1].

Une meilleure gestion des exigences promeut également la communication des effets qu'aura la mise en place du système et les changements que ce dernier provoquera.

En définitive, cette phase doit permettre à toutes les parties prenantes d'obtenir une parfaite compréhension des exigences et donner l'assurance que les personnes ayant le pouvoir de décision comprennent et approuve les exigences et la portée du produit.

2.2.2. Objets de modélisation

Causes (Rationale)

La modélisation des causes apporte une plus-value aux conflits qui peuvent survenir dans les phases décisionnelles, en gardant une trace des résolutions précédentes de ces conflits et des raisons qui justifient sa solution.

[1] (International Institute of Business Analysis, 2009)

Portée du produit

Les limites de la solution à ce niveau doivent être validées, comprises et confirmées par les parties prenantes afin de fixer pour le reste du cycle de l'IE les exigences qu'il conviendra de définir et d'implémenter[1].

Traçabilité des exigences

Depuis une exigence, plusieurs liens de traçabilité doivent être créés : vers une ou plusieurs autres exigences de même niveau d'abstraction (relation) ; en amont vers un niveau d'abstraction plus élevé et en remontant jusqu'aux causes et aux buts (dérivation) ; en aval vers un niveau d'abstraction plus profond en descendant jusqu'aux demandes de changements et aux tests (allocation)[2]. Ces liens de traçabilité permettront d'analyser l'impact des changements futurs sur le système.

[1] (International Institute of Business Analysis, 2009)
[2] (International Institute of Business Analysis, 2009)

2.3. Phase 3 : Priorisation et organisation

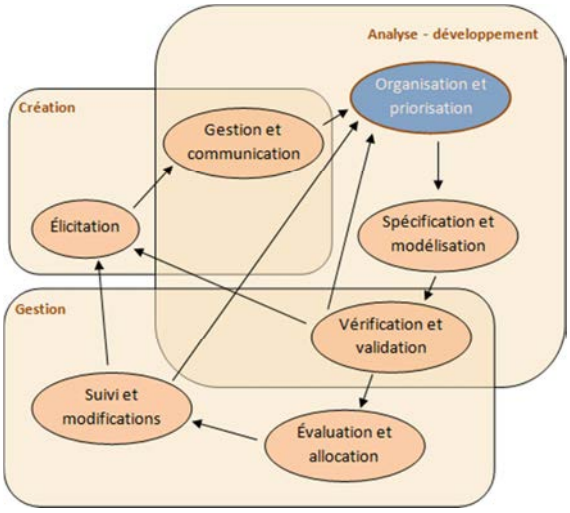

Figure V-3 - La Priorisation et l'Organisation dans le cycle de l'IE

2.3.1. Description

L'attribution des priorités des exigences traduit l'importance relative de chaque objectif et définit les packages les plus critiques à implémenter et sur lesquels les efforts doivent se concentrer[1].

L'organisation des exigences se fait par leur articulation autour de plusieurs vues sur la solution, permettant d'assurer leur compréhension, leur complétude, et leur consistance.

2.3.2. Objets de modélisation

Découpage du projet

Lors des phases d'élicitation des processus itératifs de gestion de projets, la réalisation d'une découpe permettra plus facilement de définir des priorités sur les exigences et de délimiter les packages à analyser[2].

Objectifs du système

L'analyse et la modélisation des objectifs du système permettront de vérifier qu'une exigence est prioritaire et pertinente car elle satisfait un objectif. A l'inverse, si une des parties prenantes formule une exigence qu'il est impossible de justifier au moyen des objectifs, on peut en déduire une faible priorité[3].

[1] (International Institute of Business Analysis, 2009)
[2] (International Institute of Business Analysis, 2009)
[3] (van Lamsweerde, 2001)

Priorité des exigences

Les modèles de priorité des exigences permettent d'une part de les calculer en fonction des critères exprimés par les parties prenantes, et d'autre part de les représenter visuellement.

Traçabilité des exigences

La traçabilité aval et amont des exigences permet d'étudier leur dérivation, ces modèles sont de ce fait un bon moyen de repérer le niveau d'abstraction auxquelles elles appartiennent et les packages dans lesquels il faudra les inclure.

2.4. Phase 4 : Spécification et modélisation

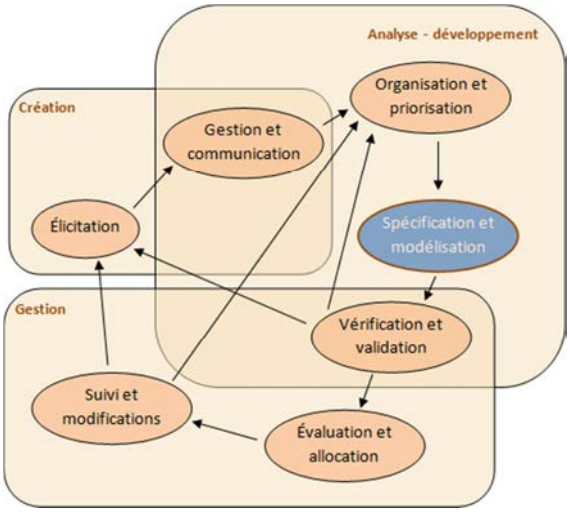

Figure V-4 - La Spécification et la Modélisation dans le cycle de l'IE

2.4.1. Description

Il s'agit de décrire les exigences, par une spécification non-formelle ou formelle, ou de les modéliser graphiquement. Cette phase est visiblement incontournable lorsque l'on souhaite utiliser les modèles pour l'ingénierie des exigences. La modélisation des concepts, des états, des flux et processus du système peut se faire également pour détailler les exigences.

Les critères qualifiant une bonne spécification sont les caractères de concision, complétude, cohérence, consistance, et de non-ambiguïté[1]. Les modèles apportent justement un état de concision grâce à la syntaxe graphique dont ils tirent parti. La complétude peut être assurée par la combinaison de plusieurs modèles pour exprimer une même exigence. Il est également utile de choisir, parmi les outils logiciels offrant des possibilités de création de modèles, ceux qui lient les objets graphiques de schémas différents car cela assure une certaine cohérence et consistance dans la nomenclature.

2.4.2. Objets de modélisation

Concepts du domaine

Les standards de modélisation des données métier, ou objets du domaine, offrent des représentations à différents niveaux de détail. Lorsqu'elle est utilisée pour l'ingénierie des exigences uniquement, la modélisation des données reste généralement d'assez faible degré de détail, sans utiliser l'ensemble de la syntaxe disponible (types d'objet, relations, principales propriétés). Cependant, un détail plus profond constitue un apport considérable à la suite de l'analyse et une contribution à la future architecture du système[2].

[1] (Wiegers, 2003)
[2] (Podeswa, 2009)

Les exigences que ces modèles permettent de spécifier sont les cardinalités existantes ou autorisées de leurs relations, et les actions que l'utilisateur doit pouvoir initier sur chaque objet métier.

Enchaînement des états

Ils sont utilisés en IE pour modéliser les exigences de réponses aux évènements, les conditions de déclenchement des activés, et les conditions de modification de l'état d'un objet.

Flux

La modélisation des flux conçoit une vue haut niveau sur les exigences en termes d'échanges de données entre les différents processus du système. Elle donne à percevoir également une vue réaliste du système existant, ou du fonctionnement espéré. La modélisation permet de spécifier également quels sont les entrées et les sorties de chaque processus.

Objectifs du système

Elle apporte une vue très haut niveau sur les exigences et les objectifs qu'ont les utilisateurs envers le système.

Processus métier

Leur modélisation permet de spécifier quels sont les processus de haut niveau d'abstraction que le système devra implémenter, et quelles en sont les principales tâches.

Scénarios d'utilisation

Ces modèles spécifient les exigences de fonctionnement du système pour chaque cas d'utilisation, d'une perspective de l'utilisateur. Ils permettent de représenter les scénarios normaux et alternatifs, ainsi que leurs préconditions et postconditions respectives.

Scénarios système

Ces modèles expriment les exigences sur le fonctionnement interne du système pour chaque scénario d'utilisation ou processus métier.

2.5. Phase 5 : Vérification et validation

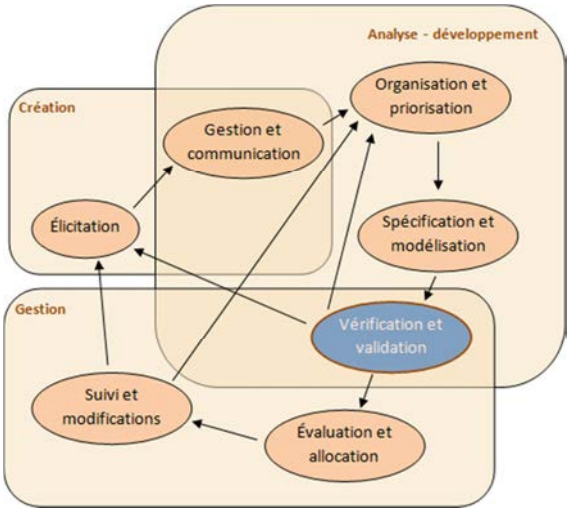

Figure V-5 - La Vérification et la Validation dans le cycle de l'IE

2.5.1. Description

A ne pas confondre avec le terme « Final V&V » utilisé régulièrement en anglais dans le domaine de l'analyse de projets IT, où V&V signifie également *Verification and Validation* mais dans une perspective de différentes passes de tests. Nous avons choisi de regrouper les tests dans la phase « suivi ».

La vérification est l'assurance que les spécifications des exigences respectent un standard de qualité venant consolider l'efficacité de leur utilisation dans la suite du projet. Les exigences doivent être définies correctement et toute l'information nécessaire doit être fournie afin qu'elles puissent être validées convenablement par les parties prenantes[1].

La validation est l'acceptation des exigences en fonction de leur apport d'une valeur au business, de leur complétude des besoins métier et de leur réponse aux besoins des parties prenantes.

2.5.2. Objets de modélisation

Objectifs du système

La modélisation des objectifs du système permettra de valider que l'ensemble des exigences utilisateurs ont été découvertes.

Processus métier

Ces modèles permettent aux parties prenantes de valider que les processus métier ont été correctement et complètement compris et spécifiés.

[1] (International Institute of Business Analysis, 2009)

Scénarios d'utilisation

Offrant une vue facile à lire et interpréter pour les personnes non-techniques, elle permet de faire vérifier par les parties prenantes que la réalisation d'un objectif se fera selon le fonctionnement attendu.

Traçabilité des exigences

Ces modèles peuvent montrer que chaque exigence répond à un besoin utilisateur, grâce à la traçabilité amont.

2.6. Phase 6 : Evaluation et Allocation

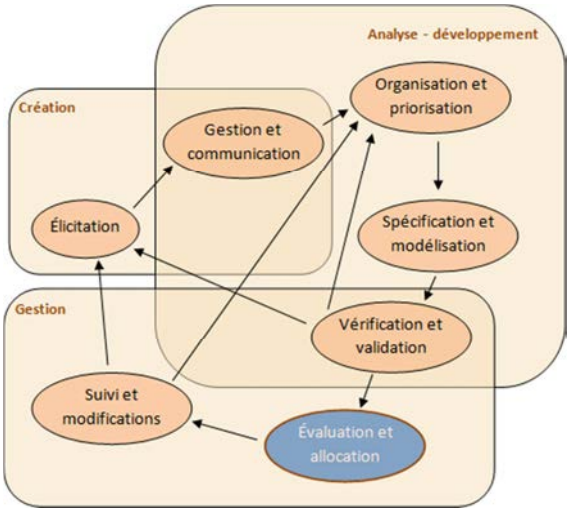

Figure V-6 - L'Evaluation et l'Allocation dans le cycle de l'IE

2.6.1. Description

La solution proposée doit vérifier les exigences des parties prenantes et apporter une valeur ajoutée au métier. Cette sixième phase du cycle concerne la réalisation de cette tâche mais encore celle de l'organisation des exigences en packages séparés verticalement en fonction de leurs objectifs et horizontalement en fonction de leur niveau d'abstraction (métier, utilisateurs, système)[1].

Ces packages ou composants de solution permettront de définir des « releases » afin de maximiser les bénéfices et minimiser les coûts.

2.6.2. Objets de modélisation

Découpage du projet

Les modèles de découpage du projet constituent le meilleur moyen d'avoir une vision sur les packages d'exigences, et de repérer éventuellement des conflits ou des liens immuables.

Priorité des exigences

L'analyse des priorités est la principale activité qui permettra à l'équipe technique d'attribuer une importance aux objectifs, et donc indirectement aux packages d'exigences.

Processus métier

Ces modèles permettent aux parties prenantes de valider que les processus métier ont été correctement et complètement compris et spécifiés.

[1] (Podeswa, 2009)

Scénarios système

Le fonctionnement interne du système, les interactions entre les utilisateurs et les différents composants de ce système doivent être validés par les parties prenantes et leur modélisation rend cette tâche plus pratique.

2.7. Phase 7 : Suivi et modifications

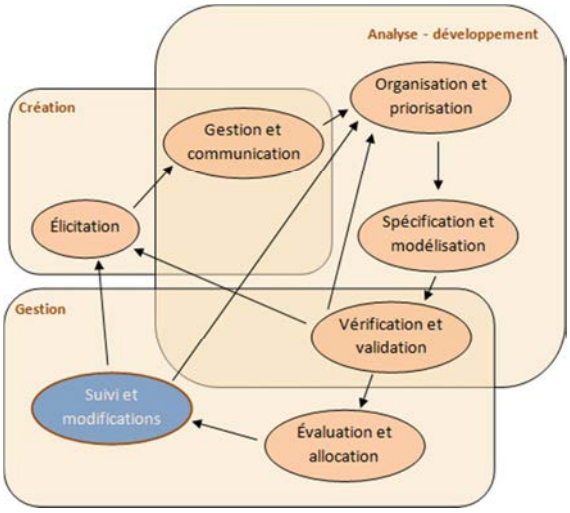

Figure V-7 - Le Suivi et les Modifications dans le cycle de l'IE

2.7.1. Description

Le suivi concerne la création des cas de tests, leur gestion, et l'assurance que le comportement du système correspond aux exigences utilisateur et système.

En raison des incohérences et/ou inconsistances soulevées par les tests dans les exigences, c'est lors de cette étape que les sources problématiques ont l'opportunité d'être identifiées et découvertes [1].

Les modifications à apporter au système doivent être gérées en évaluant au préalable l'impact qu'elles provoqueront en son sein, en détectant des potentiels conflits qu'elles génèreront.

2.7.2. Objets de modélisation

Enchaînement des états

La couverture des tests est vérifiable via la modélisation des états. En effet, les tests doivent couvrir tous les états possibles et parcourir l'ensemble des transitions afin de valider le comportement attendu de l'exigence initiale[2].

Scénarios d'utilisation

Les scénarios des utilisateurs sont une assise adéquate pour la mise au point des tests d'acceptation. Ils certifient que le système répond de la manière attendue et qu'une précondition définie donne bien l'issue spécifiée dans la postcondition du scénario correspondant.

[1] (International Institute of Business Analysis, 2009)
[2] (Podeswa, 2009)

Scénarios système

Elle peut être utilisée à un bas niveau, dans les phases de construction des tests d'intégration. Cette tâche devant suivre précisément les spécifications des processus, la modélisation formelle de ces derniers constitue un fondement solide pour la création et validation des tests[1].

La modélisation des processus peut également être utilisée pour analyser l'impact d'une modification d'un processus sur un système existant.

Traçabilité des exigences

Les techniques de gestion de la traçabilité sont utilisées afin de relier les exigences aux différents cas de tests, et aux historiques de changement.[2]

[1] (Podeswa, 2009)
[2] (Alexander, et al., 2009)

3. Récapitulatif

Tableau V-A - Récapitulatif des exigences pour chaque phase

	Phase 1 **Élicitation**	Phase 2 **Gestion et communication**	Phase 3 **Priorisation et organisation**	Phase 4 **Spécification et modélisation**	Phase 5 **Vérification et validation**	Phase 6 **Evaluation et Allocation**	Phase 7 **Suivi et modifications**
Causes (Rationale)		x					
Concepts du domaine	x			x			
Découpage du projet			x			x	
Enchaînement des états	x			x			x
Flux				x			
Objectifs du système	x		x	x	x		
Parties prenantes	x						
Portée du produit	x	x					
Priorité des exigences			x			x	
Processus métier	x			x	x	x	
Scénarios d'utilisation	x			x	x		x
Scénarios système				x		x	x
Traçabilité des exigences	x	x	x		x	x	x

Légende :

x OM utilisable dans la phase définie

Nous voulons souligner les apports de ce récapitulatif. Il est possible de déduire de cette synthèse que les OM les plus utiles au sein du cycle de vie de l'IE sont :

- La modélisation de la traçabilité des exigences, avec une utilisation dans 6 phases sur 7.
- La modélisation des objectifs du système,
- La modélisation des processus métier,
- La modélisation des scénarios d'utilisation, toutes les trois à égalité avec une utilisation dans 4 des 7 phases.

De manière croisée, les phases qui tireront un meilleur parti de l'utilisation des modèles sont :

- L'Elicitation
- La Spécification et la modélisation
- L'Evaluation et l'allocation

CHAPITRE VI.

LES MODÈLES ET LES OUTILS D'IE

1. Introduction

Dans les chapitres précédents, nous avons établi différentes catégorisations des concepts qui peuvent être schématisés dans les modèles. Nous avons dénommé ces concepts « objets de modélisation ».

Dans un premier temps, nous avons mis en avant chacune des perspectives qu'ils pouvaient apporter. Pour chacune d'entre elles, nous avons identifié l'OM le plus adéquat.

Ensuite, nous avons indiqués quels étaient les standards qui convenaient pour la spécification et la découverte des exigences fonctionnelles, puis des exigences non-fonctionnelles. Certains OM sont utiles au sein de l'IE mais ne sont pas adaptés pour la spécification, ni la découverte des exigences.

En troisième lieu, les besoins de spécification des exigences à un certain niveau d'abstraction ont été étudiés afin de présenter les OM propices à chacun d'eux.

Enfin, la dernière classification s'est opérée sur les phases du cycle de l'IE. Chacune d'elles a des besoins spécifiques et propres aux activités qu'elle requiert, nous avons donc montré comment les OM pouvaient aider dans ces activités.

Choix des modèles

Ces classifications pourront aider l'analyste à identifier les OM qui pourront répondre à ses exigences et les standards qui seront plus adaptés aux particularités dont il a besoin. En ayant découvert les OM qui le serviront au mieux dans ses tâches, il pourra ainsi limiter sa recherche aux outils qui permettent leur création.

Gestion vs. Modélisation des exigences

Certains outils d'ingénierie des exigences ne sont spécifiques qu'à la gestion des exigences, d'autres permettent également de les modéliser. Bon nombre d'outils existent également et ne sont dédiés qu'à la création de modèles.

Il est crucial, lorsque l'on désire utiliser les modèles pour la spécification des exigences, de pouvoir lier la création de ces modèles à la gestion des exigences qui en dépendent. C'est pourquoi nous avons identifié une série de fonctionnalités des outils nécessaires à établir cette liaison.

Ces exigences d'outils d'IE sont exposées sous la forme d'un petit « framework » dédié spécifiquement au choix d'un outil pour la modélisation.

2. Grille de sélection des OM et des standards de modélisation

Tableau VI-A - Grille de sélection des modèles selon différents critères

	Standard	Modèle	Fonctionnelles	Non fonctionnelles	Métier	Utilisateur	Système	Personnes	Données	Fonctionnement	Comportement	Objectifs	Suivi et modifications	Evaluation et Allocation	Vérification et validation	Spécification et modélisation	Priorisation et organisation	Gestion et communication	Élicitation
Causes (Rationale)	GSN	Notation GSN	M	DM															
		Modèle de causes		DM				x				x						x	
	i*	Strategic Rationale	DM	DM															
	KAOS	Modèle de buts		DM															
Concepts du domaine	UML	Diagramme de classes	DM	DM	x	x													
		Schéma Entités-Associations	DM	DM	x				x							x			x
	RML	Diagramme de données métier	DM	DM	x														
Découpage du projet	UML	Diagramme de Package																	
		Structure de découpage du projet										x		x			x	x	
	RML	Arbre des fonctions	D																
Enchaînement des états	UML	Diagramme d'états-transitions	DM	DM	x		x		x		x		x		x	x		x	x
	RML	Diagramme d'état	DM	DM	x														
Flux	SA	Diagramme des flux de données	DM						x	x	x					x			x
	UML	Diagramme de communication	DM		x														
Objectifs du système	UML	Diagramme des cas d'utilisation	DM			x													
	i*	Strategic Dependency	DM	DM	x	x		x				x		x	x	x	x	x	x
	RML	Modèle d'objectifs métier	D	M	x														
Parties prenantes		Modèle en oignon																	
		Matrice d'influence						x											
	RML	Diagramme de l'organisation																	
Portée du produit	SA	Diagramme de contexte	DM	D	x		x												
		Graphe de décomposition fonctionnelle									x	x						x	x
	RML	Carte de l'écosystème	DM	D	x		x												
Priorité des exigences		Processus d'analyse hiérarchique		D															
		Matrice QFD										x		x			x		x
		Calcul du retour sur investissement		D															
Processus métier	UML	Diagramme d'activité	DM	D	x														
	BPMN	Diagramme de processus métiers	DM	D	x			x				x		x	x	x			x
	RML	Flux de processus	DM	D	x														
Scénarios d'utilisation		Prototypage	DM	DM		x													
	UML	Diagramme d'activité	DM	DM	x	x		x		x	x		x		x	x			x
	RML	Enchaînement des interfaces utilisateur	DM	DM		x													
Scénarios système	UML	Diagramme de séquence	DM	D		x	x			x		x	x	x		x			x
	RML	Flux de système	DM	D		x	x												
Traçabilité des exigences		Table de traçabilité	M	M															
		Ensemble de matrices de traçabilité	M	M				x				x	x	x	x		x	x	x

3. Framework pour l'aptitude des outils d'IE à la modélisation

Pour la réalisation de ce framework, nous nous sommes inspirés d'une partie de l'étude de Seilevel[1] ainsi que de l'identification de fonctionnalités dans des outils existants[2].

1. L'outil dispose de capacités de modélisation :

Pour un OM défini, l'outil offre…

- La possibilité de générer automatiquement des modèles à partir d'informations présentes dans l'outil (données textuelles, liaisons, utilisateurs, …)
- Une interface de création manuelle graphique de modèles
- La possibilité à plusieurs utilisateurs de modifier un modèle simultanément
- La possibilité de verrouiller les modifications sur un modèle
- Le choix entre plusieurs standards du marché
- Un standard défini uniquement

2. L'outil offre d'intégrer des modèles externes :

Pour un OM défini, l'outil peut importer des modèles externes…

- Par connexion synchronisée avec un logiciel externe (la modification du modèle dans l'outil de modélisation se reflète dans l'outil de gestion)
- Sous forme d'objets assemblés (décryptage de types de fichiers propres à un ou plusieurs outils de modélisation)
- Sous forme d'image

3. L'outil permet la liaison de données au modèle :

L'outil permet de…

- Lier, à un objet d'un modèle, une exigence/un utilisateur de l'outil/un acteur du produit
- Lier, à une zone d'un modèle, une exigence/un utilisateur de l'outil/un acteur du produit
- Lier, à l'entièreté d'un modèle, une exigence/un utilisateur de l'outil/un acteur du produit
- Lier un élément d'un modèle à un élément d'un autre modèle
- Lier un élément d'un modèle à un autre modèle
- Maintenir les liaisons lors de la modification du modèle

[1] (Beatty, et al., 2011)
[2] Les outils testés sont Microsoft Visio et Visual Paradigm for UML

4. L'outil permet la validation du modèle :

L'outil permet de…

- Créer des utilisateurs de lecture seule ayant des droits de validation des modèles
- Approuver/Rejeter séparément des objets d'un modèle
- Approuver/Rejeter l'entièreté d'un modèle
- Apposer des commentaires aux approbations/rejets
- Apposer des commentaires de validation sur une zone d'un modèle
- Apposer des commentaires de validation sur l'entièreté d'un modèle

5. L'outil offre la génération de données à partir d'un modèle :

L'outil permet de…

- Générer des exigences à partir d'un modèle
- Créer des utilisateurs de l'outil à partir d'un modèle de personnes

6. L'outil permet l'impression des modèles :

L'outil permet de…

- Imprimer directement un modèle
- Intégrer automatiquement les modèles liés à une exigence dans un document d'exigences
- Sélectionner les modèles à imprimer dans un document d'exigences

CONCLUSION

Le présent travail a été réalisé dans le but de l'achèvement des études de master en sciences informatiques à l'université de Namur. Il clôture deux années de cours du soir au fil desquelles des notions d'ingénierie du logiciel, de gestion de projet et d'analyse orientée objet ont été acquises. L'étude de cas réalisée dans le cadre du laboratoire du cours d'ingénierie du logiciel, a été particulièrement appréciée grâce aux nouveaux concepts qu'elle a permis d'aborder en IE. Le projet commençait en effet, par une analyse conséquente des exigences de l'étude de cas, au cours de laquelle il a été proposé d'émettre des spécifications par le biais de modèles UML. Nous avons été intrigué par les options qu'offraient ces derniers au cours de l'ingénierie des exigences. Un poste d'analyste nous a été proposé conjointement aux choix du thème de mémoire, ce qui nous a dirigé vers le présent sujet. L'objectif recherché était de profiter de cette opportunité pour augmenter nos connaissances dans le domaine de l'IE.

En ce qui concerne l'ingénierie des exigences, nous avons en premier découvert qu'elles sont en constante évolution au cours de l'étude d'un produit. Elles suivent un processus qui peut être long et volumineux (par exemple dans un gros projet suivant une gestion en cascade) ou assez court et moins conséquent (par exemple dans un projet Agile ou suivant une gestion itérative). Beaucoup d'auteurs[1] tentent de fournir la liste des étapes de l'évolution des exigences au cours du projet, selon le point de vue qu'ils tirent de leur propre expérience. Nous avons étudié et recoupé les tentatives issues de ces différents écrits, afin d'élaborer un cycle de vie qui puisse être utile dans notre comparaison. Le cycle conceptualisé comporte les sept phases suivantes : Elicitation (découvrir les exigences), Gestion et communication (assurer leur compréhension), Priorisation et organisation (cibler les efforts), Spécification et modélisation (caractériser les exigences), Vérification et Validation (vérifier la qualité et les objectifs), Evaluation et Allocation (vérifier la solution et déterminer les livrables) et enfin Suivi et modifications (tester et gérer les changements).

Ce cycle a été mis au point pour cerner les phases du projet où les modèles apportaient leur utilité. Afin de pouvoir également présenter les apports des modèles sous d'autres perspectives, nous avons relevé deux classifications supplémentaires communément acceptés. Ainsi, ont été introduits, les degrés de fonctionnalité des exigences (fonctionnel et non-fonctionnel) et leurs niveaux d'abstraction (métier, utilisateur et système).

Le nœud de l'étude étant la comparaison des outils logiciels utilisables en ingénierie des exigences, un aperçu de ce qui existait en la matière a été exposé. Nous avons montré trois exemples d'approches de comparaison sous la forme de liste de fonctionnalités, dénommée « *framework* ». Parmi ceux étudiés, seul un[2] fait référence à plusieurs capacités ayant trait aux modèles. Un second[3] se contente de vérifier que l'outil puisse spécifier les exigences de manière formelle ou semi formelle tandis que le dernier[4] n'en fait pas mention. Ce manquement a suscité notre questionnement et permis de mieux cibler notre recherche.

[1] (Institute of Electrical and Electronics Engineers, Inc., 1998), (International Institute of Business Analysis, 2009), (Wiegers, 2003), (Robertson, et al., 2006) et (Beatty, et al., 2012)
[2] (Beatty, et al., 2011)
[3] (Matulevicius, 2005)
[4] (INCOSE Requirements Working Group, 2010)

Outre les recherches visant expressément la comparaison d'outils, certains ouvrages plus généraux à l'IE proposent une approche de sélection mais sans fournir de framework. Par contre, ces ouvrages mettent tous en avant l'un ou l'autre modèle en exprimant ses utilités dans l'IE. Nous avons étudié celles-ci en dressant un inventaire des standards de modélisation rencontrés et en relevant leurs avantages et inconvénients. Certains ont plus de 40 ans d'âge mais sont toujours utilisés par bon nombre d'analystes, tandis que d'autres sont plus récents et apportent chacun une complémentarité intéressante en regard de tous les standards existants.

Les standards rencontrés, présentés dans la première partie des annexes, ont été classifiés en treize groupes en fonction du concept qu'ils permettent de représenter. Ces groupes, que nous avons nommés « Objets de Modélisation », sont les suivants : Causes, Concepts du domaine, Découpage du projet, Enchaînement des états, Flux, Objectifs du système, Parties prenantes, Portées du produit, Priorité des exigences, Processus métier, Scénarios d'utilisation, Scénarios système et Traçabilité des exigences.

Chacun d'eux apporte son utilité à l'IE, certains pour la spécification, d'autres pour la recherche d'exigences ou encore pour leur gestion. En constatant cette variété d'apports, nous avons trouvé pertinent de les classifier dans le but de mettre les OM en comparaison. Plus précisément, nous avons défini, en nous inspirant de plusieurs sources existantes[1], des perspectives que l'analyste pouvait nécessiter dans son processus d'IE. Adaptées des classes proposées par ces auteurs, nous avons reconnu et préféré un ensemble de cinq perspectives sur les exigences percevables par les OM : Les Objectifs, Le Comportement, le Fonctionnement, les Données et les Personnes. Ces perspectives donnent à l'analyste, grâce à la comparaison réalisée, la possibilité de choisir un ou plusieurs OM pour ses objectifs propres.

Ensuite, pour chaque objet de modélisation, nous avons comparé les standards au moyen des deux critères énoncés plus tôt (degré de fonctionnalité et niveau d'abstraction). Cette comparaison a pour vocation d'aider l'analyste à choisir le standard qui lui conviendrait plus adéquatement.

Les précédentes comparaisons étant centrées sur la manière d'utiliser les modèles, nous avons jugé intéressant de comparer ensuite les moments où ils interviennent. Plus spécifiquement, ce second apport personnel a été d'identifier les OM qui pouvaient offrir davantage d'efficacité lors d'une phase déterminée du cycle de l'IE. Nous avons expliqué qu'il appartient aux analystes de choisir les phases qu'ils voudront agrémenter par l'utilisation des modèles. En effet, chacun a sa manière propre de travailler, son processus mis en place et bien défini.

Cette dernière comparaison montre que chaque phase peut tirer parti de plusieurs OM, mais qu'il en existe trois où ils sont manifestement plus importants. La première, assez logiquement, est la phase de Spécification et modélisation. A égalité, avec un intérêt apporté par 9 OM sur 13, se trouve la phase d'Elicitation. Vient ensuite en troisième position, la phase d'Evaluation et allocation. De manière croisée, les OM les plus profitables sont la Traçabilité des exigences, les Processus métier et les Objectifs du système.

[1] (Object Management Group, Inc, 2012), (Pohl, et al., 2011) et (Beatty, et al., 2012)

De la mise en commun des données de ces différentes comparaisons est apparue une grille ayant pour but d'aider l'analyste dans la sélection des OM et des standards qui lui seront appropriés. Ce choix de modèle est ensuite complété par un framework de comparaison des fonctionnalités d'un outil en termes de support des modèles.

Le choix de l'outil est la dernière, mais non moins importante, étape de comparaison. Outre le fait de pouvoir modéliser un OM défini selon le standard choisi, l'outil doit pouvoir éventuellement importer des modèles externes. Il doit permettre également de lier des données (exigences ou autres) aux modèles, offrir des possibilités de validation d'un modèle par les parties prenantes, avoir la capacité de générer des données (exigences ou autres) à partir d'un modèle, et enfin d'introduire ceux-ci dans les documents de spécifications des exigences.

En définitive, nous avons complété les approches de comparaison d'outils d'IE de deux manières. La première consiste en une grille de sélection des modèles en fonction des besoins de l'utilisateur en termes d'amélioration du cycle de l'IE, du besoin spécifique d'une perspective sur les exigences, et enfin de la nécessité de les découvrir ou les modéliser selon deux niveaux que sont la fonctionnalité et l'abstraction. La seconde énumère un ensemble d'exigences fonctionnelles d'outils d'IE qui permettra à l'utilisateur de mieux évaluer les aptitudes de ceux-ci face aux modèles.

Nous reconnaissons que ce framework de sélection d'outils doit comporter des manquements. En effet, trop peu d'outils ont été analysés que pour répertorier une gamme complète d'exigences fonctionnelles. De plus, l'élaboration de cette liste ne s'appuie pas sur une étude conséquente mais sur un seul ouvrage[1] existant. Si l'opportunité nous était donnée de poursuivre cette recherche, nous envisagerions en premier de compléter cette liste en comparant un nombre pertinent d'outils d'IE.

Au niveau de l'étude des modèles, nous tenterions d'être plus exhaustif dans les standards de modélisation courants, en commençant par ceux que nous avons déjà mis en lumière dans le présent écrit. Une autre amélioration possible devrait éviter à l'utilisateur de rechercher lui-même l'outil qui lui permette de modéliser selon le standard opté. Pour ce faire, il serait envisageable et avantageux de créer une base de données répertoriant les différents outils disponibles sur le marché.

Ce mémoire, grâce aux diverses comparaisons opérées, nous a permis de retirer les différents fruits de l'utilisation des modèles en IE. La plus grande découverte a été le langage RML® qui a été mis au point expressément pour la modélisation des exigences. Il apporte de nombreux modèles complémentaires que nous serions très intéressés d'utiliser dans de futures études de cas.

Nous espérons que les résultats du présent travail, à visée essentiellement pratique, porteront bénéfice à un analyste recherchant à instaurer l'utilisation des modèles dans son ingénierie, et qu'ils pourront favoriser davantage l'attrait pour ceux-ci dans la spécification et la recherche d'exigences.

[1] (Beatty, et al., 2011)

BIBLIOGRAPHIE

Ahmad, Azeem, Goransson, Magnus et Shahzad, Aamir. 2010. Limitations of the Analytic Hierarchy Process Technique with Respect to Geographically Distributed Stakeholders. *Engineering and Technology.* 2010, 69.

AIAC. 2010. Cours d'analyse multicritère d'aide à la décision. s.l. : Académie Internationale de l'Aviation Civile, 2010.

Alexander, Ian and Beus-Dukic, Ljerka. 2009. *Discovering Requirements : How to Specify Products and Services.* s.l. : Wiley, 2009.

Babou, S. 2008. What is Stakeholder Analysis? *The Project Management Hut.* [En ligne] 12 Mars 2008. [Citation : 26 Juillet 2012.] http://www.pmhut.com/what-is-stakeholder-analysis.

Beatty, Joy et Chen, Anthony. 2012. *Visual Models for Software Requirements : An RML® Handbook.* s.l. : Microsoft Press, 2012.

Beatty, Joy et Ferrari, Remo. 2011. *How to Evaluate and Select a Requirements Management Tool.* s.l. : Seilevel, 2011.

Bourne, Lynda et Walker, Derek H.T. 2006. Using a visual tool to study stakeholder influence. *The Project Management Journal.* 2006, Vol. 37, N° 1.

Buzan, Tony. 1976. *Use Both Sides of Your Brain.* s.l. : Dutton paperback, 1976.

Cabinet Office. 2011. *Glossaire français ITIL® v1.1.* [éd.] © Crown Copyright 2011. s.l. : Cabinet Office, 2011.

Davis, Guy, Zannier, Carmen et Geras, Adam. 2001. *QFD for Software Requirements Management.* s.l. : MSc of Software Engineering Program, 2001.

Grunske, Lars, Reussner, Ralf H. et Plasil, Frantisek. 2010. *Component-Based Software Engineering.* s.l. : Springer, 2010.

INCOSE Requirements Working Group (2). 2010. Requirements Management Tools Survey Responses. *International Council on Systems Engineering.* [En ligne] 08 Août 2010. [Citation : 24 Mai 2012.] http://www.incose.org/ProductsPubs/products/INCOSERMToolSurveyConsolidatedResults.xls.

INCOSE Requirements Working Group. 2010. Requirements Management Tools Survey. *International Council on Systems Engineering.* [En ligne] 08 Août 2010. [Citation : 24 Mai 2012.] http://www.incose.org/ProductsPubs/products/rmsurvey.aspx.

Institute of Electrical and Electronics Engineers, Inc. 1998. IEEE Guide for Developing System Requirements Specifications. *IEEE Std 1233a.* 1998. 1233. ISBN 0-7381-1515-0.

International Institute of Business Analysis. 2009. *Business Analysis Body of Knowledge, Version 2.0 (BABOK Guide).* s.l. : International Institute of Business Analysis, 2009.

Kerzazi, N. 2010. *Exigences et spécifications du logiciel.* École Polytechnique de Montréal. Montreal, Québec, Canada : s.n., 2010. Cours.

Matulevicius, Raimundas. 2005. *Process Support for Requirements Engineering.* Trondheim : Norwegian University of Science and Technology, 2005.

McCaffrey, James. 2005. Test Run: The Analytic Hierarchy Process. *MSDN Magazine.* 2005, June.

Merts, Joseph P. 2009. The Psychology of Notation. *QFD Online.* [En ligne] 10 Janvier 2009. [Citation : 19 Juillet 2012.] http://www.qfdonline.com/archives/the-psychology-of-notation/.

Object Management Group, Inc. 2012. Introduction to OMG's Unified Modeling Language™ (UML®). *Wikipedia.* [En ligne] 24 Août 2012. [Citation : 24 Août 2012.] http://www.omg.org/gettingstarted/what_is_uml.htm.

Origin Consulting (York) Ltd. 2011. *GSN Community Standard, version 1.* 2011.

Petit, Michaël. 2008-2009. *E-Business - IHDC2214 Syllabus.* Namur : Facultés Universitaires Notre-Dame de la Paix, 2008-2009.

Podeswa, Howard. 2009. *The Business Analyst's Handbook.* s.l. : Course Technology, Cengage Learning, 2009.

Pohl, Klaus et Rupp, Chris. 2011. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam.* s.l. : O'Reilly Media, Inc., 2011.

Robertson, Suzanne and Robertson, James. 2006. *Mastering the Requirements Process, Second edition.* s.l. : Addison-Wesley, Pearson Education, 2006.

Seilevel. 2012. Leadership Team. *Seilevel Requirements Defined.* [En ligne] 2012. [Citation : 20 08 2012.] http://www.seilevel.com/who-we-are/leadership/.

—. 2011. *RML Quick Reference.* s.l. : Seilevel LP, 2011.

Thompson, Rachel. 2011. Stakeholder Analysis: Winning Support for Your Projects. *MindTools.* [En ligne] 02 Mars 2011. [Citation : 26 Juillet 2012.] http://www.mindtools.com/pages/article/newPPM_07.htm.

van Lamsweerde, Axel. 2001. Goal-Oriented Requirements Engineering: A Guided Tour. [éd.] Département d'Ingénierie Informatique. s.l. : Université Catholique de Louvain, 2001.

Wiegers, Karl E. 2009. *More About Software Requirements: Thorny Issues and Practical Advice.* s.l. : Microsoft Press, 2009.

—. 2003. *Software Requirements, Second edition.* s.l. : Microsoft Press, 2003.

Wikipedia - Matrice QFD. 2012. *Wikipedia.* [En ligne] 06 Février 2012. [Citation : 21 Juillet 2012.] http://fr.wikipedia.org/wiki/Matrice_QFD.

ANNEXES – DESCRIPTION DES MODÈLES

Annexe I. Causes (Rationale)

GSN : Notation GSN (Goal Structuring Notation)

Bien que ce modèle porte le nom de « Notation structurelle des buts », il entre dans notre classification des causes. En effet, les buts et hypothèses qui y sont représentés ne sont pas des « objectifs » du système tels que nous les entendons dans un autre objet de modélisation (voir Annexe VI). Les buts dans ce modèle représentent chacun un besoin pouvant être la raison d'existence d'une exigence.

Dans ce modèle, un but de départ est discuté visuellement au moyen d'objets représentant des hypothèses, des arguments, des preuves, des stratégies, des sous-buts ou des éléments de contexte. Les éléments sont agencés en cascade descendante pour arriver à des solutions[1].

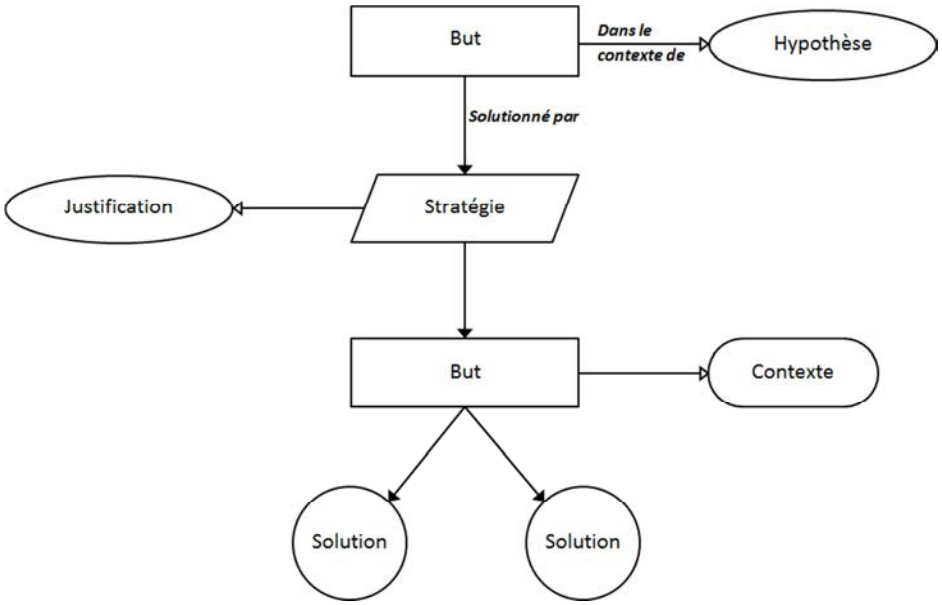

Figure VI-1 – L'ensemble de la syntaxe de la Notation GSN

Cette notation est essentiellement utilisée dans les études de sûreté dans les domaines de l'aviation, des chemins de fer, de l'aéronautique ou de la défense. Elle a pour vocation d'étudier la possiblité de mise en place d'un système[2].

i* : Strategic Rationale

La notation est très complète et propose différents concepts tels que les buts, les tâches, les ressources, les qualités[3]. Trois types de liens peuvent y être modélisés : des liens de décomposition de tâches, des liens de finalité/moyen, et enfin, des liens de contributions à une qualité[4]. Cette

[1] (Alexander, et al., 2009)
[2] (Alexander, et al., 2009)
[3] (Petit, 2008-2009)
[4] (Petit, 2008-2009)

notation analyse de manière détaillée les buts des parties prenantes. Elle met en relation les buts reconnus d'une partie prenante définie et tente d'en dériver des tâches et des exigences de qualité[1].

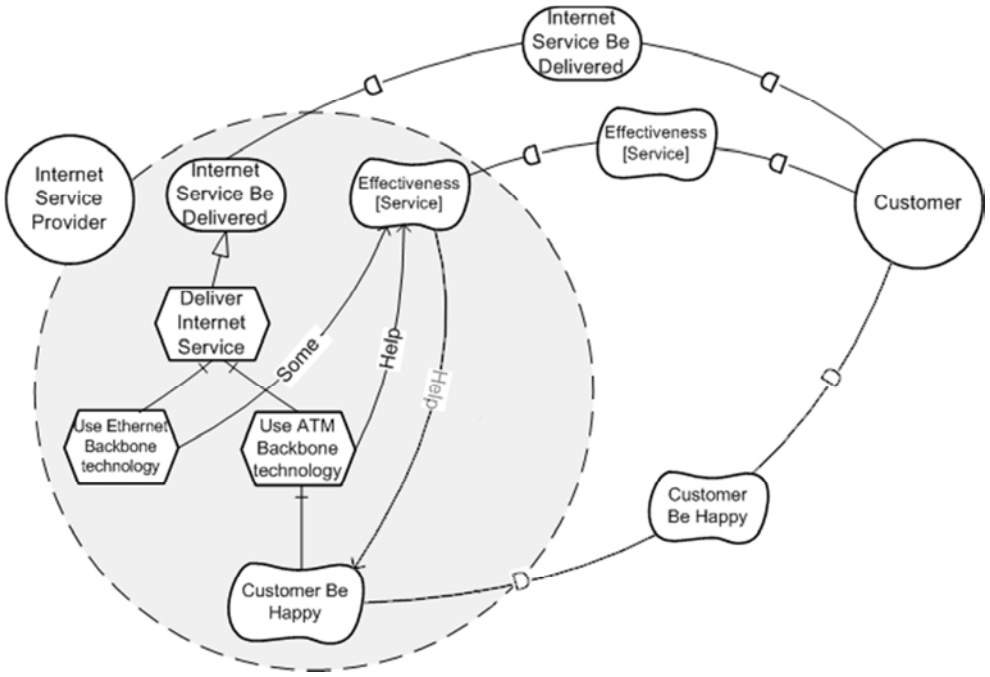

Figure VI-2 - i* Strategic Rationale[2]

Le Strategic Rationale est utilisé pour modéliser l'attrait et les motivations d'une partie prenante en regard du système à développer[3].

Modèle de causes (Rationale Model)

Ce modèle met en relation des exigences, sous la dénomination de buts techniques, avec des hypothèses et des buts métier. Les relations fléchées expriment une dépendance et peuvent être empruntées dans les deux sens : d'une exigence en remontant vers le but pour la justifier, ou d'un but en descendant vers des hypothèses afin de découvrir de nouvelles exigences[4].

[1] (Alexander, et al., 2009)
[2] Image tirée de http://istar.rwth-aachen.de/
[3] (Petit, 2008-2009)
[4] (Alexander, et al., 2009)

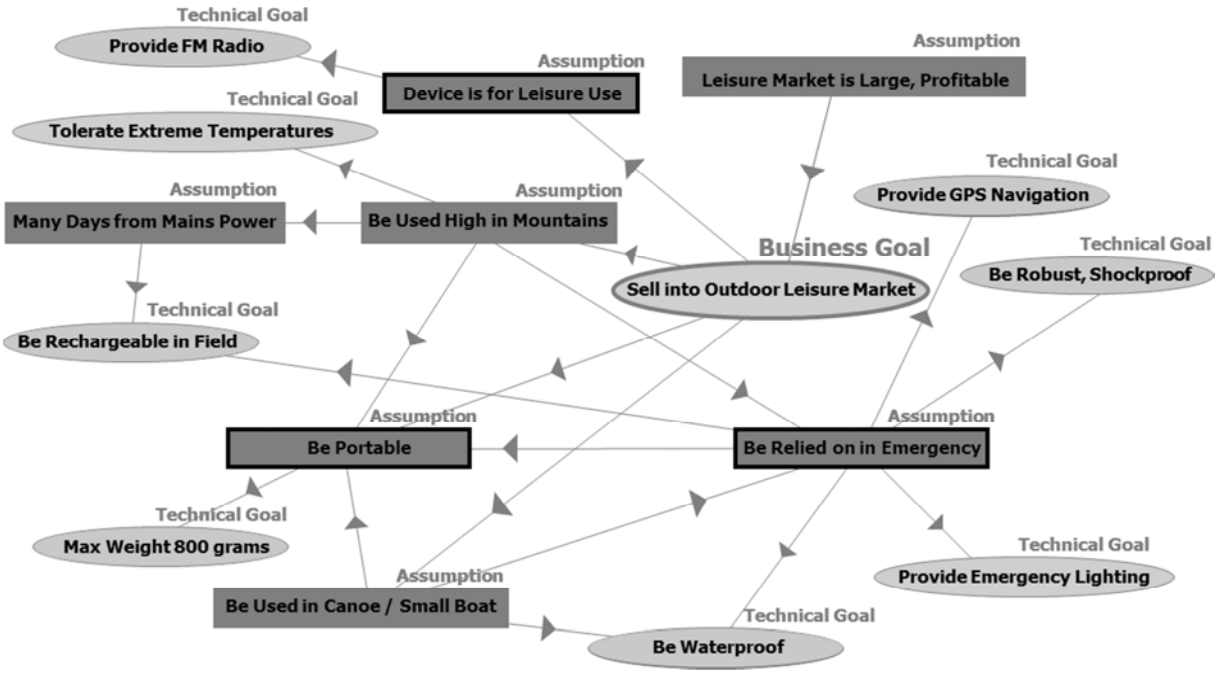

Figure VI-3 - Modèle de causes (Alexander, et al., 2009)

KAOS : Modèle de buts (Goal Model)

Ce modèle émet en relation logique de décomposition d'un premier but. La nomenclature n'est pas très riche mais permet d'identifier visuellement les potentiels conflits, ou les « synergies »[1]

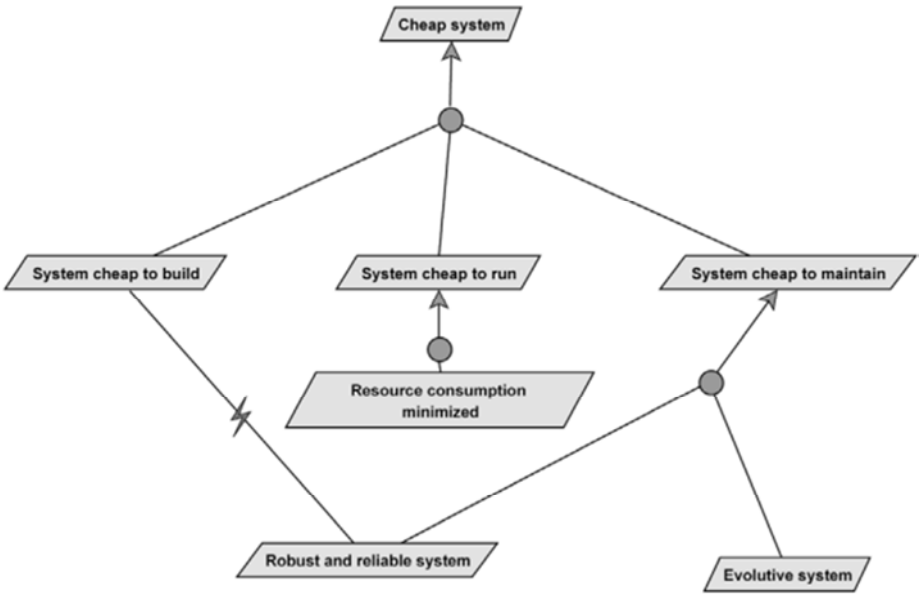

Figure VI-4 – Modèle de buts

[1] (Alexander, et al., 2009)

Annexe II. Concepts du domaine

UML : Diagramme de classes (Class Diagram)

Un diagramme de classes peut être utilisé en IE pour représenter une perspective métier des différents concepts. Les classes représentent les catégories d'objets métier. Le diagramme permet de modéliser leurs noms, leurs attributs, les actions (cas d'utilisation) qu'il est possible de leur faire faire, leurs relations, leurs dérivations, et leurs règles de multiplicité appelées cardinalités[1]. La montre un exemple de diagramme de classes comportant chacun de ces éléments.

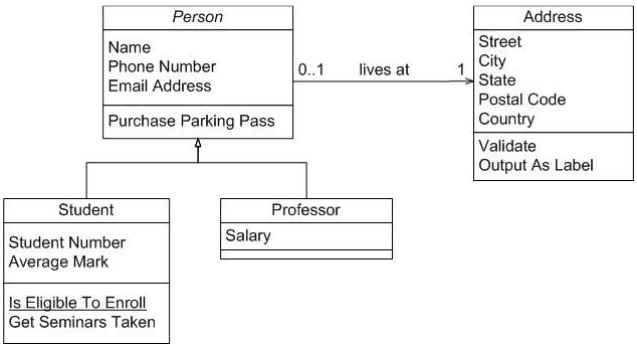

Figure VI-5 - Différents éléments d'un diagramme de classe[2]

Avec la propagation de l'*orienté objet*, qui a été majoritairement adopté déjà dans les langages de programmation et qui s'étend également à la modélisation des systèmes, le diagramme de classes est aujourd'hui fort utilisé pour la représentation des concepts du système[3].

Il a cependant l'inconvénient d'être fort imposant si l'on veut absolument être exhaustif et utiliser toutes ses possibilités. De plus, il a été observé que sa lecture et sa compréhension par des néophytes s'avère plus difficile que celle d'autres modèles tels qu'un ERD (voir point -) ou un BDD (voir point -).

Schéma Entités-Associations (Entity-Relationship Diagram)

Le schéma ERD (« *Entity Relationship Diagram* ») est énormément utilisé pour la modélisation des bases de données, cependant, il permet une très bonne appréhension des entités qui font partie d'un système[4]. Il est utilisé en ingénierie des exigences pour représenter des groupes logiques d'informations du domaine métier, ainsi que leurs interconnexions. Le modèle aide l'analyste à comprendre et communiquer les structures des composants de données[5].

[1] (Podeswa, 2009)
[2] Image tirée de http://www.agilemodeling.com/images/models/classDiagramInheritance.jpg
[3] (Alexander, et al., 2009)
[4] (Podeswa, 2009)
[5] (Wiegers, 2003)

Les éléments pouvant y être représentés sont les entités, leurs noms et leurs principaux attributs, les associations qui les relient, et les cardinalités de ces liaisons. La Figure VI-6 montre un exemple d'un schéma ERD.

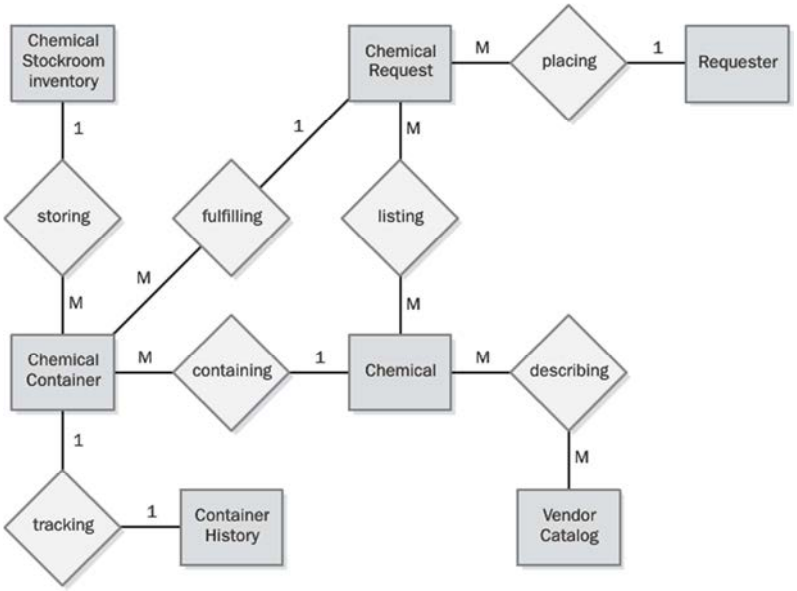

Figure VI-6 - Schéma entités-associations (Wiegers, 2003)

Bien qu'il s'agisse à ce niveau de modéliser le problème, c'est-à-dire les données du système, le schéma ERD peut servir de base aux architectes logiciels pour la réalisation d'un schéma de base de données[1].

Il existe deux types de notation pour le schéma qui se différencient par la représentation des cardinalités. Le modèle de la Figure VI-6 les représente par un chiffre et un M en anglais (pour « *Many* ») ou un N en français. La notation *Crow's Foot* décrite dans la Figure VI-7 utilise un symbole pour différencier les multiplicités[2].

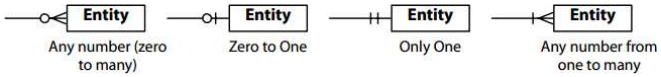

Figure VI-7 - La notation Crow's Foot (International Institute of Business Analysis, 2009)

Nous préférerons le modèle à cardinalités numériques car il permet, si l'exigence est connue, d'y représenter la multiplicité exacte de l'association[3].

[1] (International Institute of Business Analysis, 2009)
[2] (Podeswa, 2009)
[3] (Wiegers, 2003)

RML : Diagramme de données métier (Business Data Diagram)

Ce modèle présente les relations entre les objets métier. Il permet de noter les données d'après la manière dont elles sont perçues par les utilisateurs, et leur façon réelle de les créer, de les manipuler et de les enregistrer[1].

Le diagramme de données métier permet de représenter les éléments de données, leurs relations et leurs cardinalités. Il est très proche du schéma ERD vu précédemment, mais dispose d'une notation moins riche ; en effet, il a été conçu uniquement pour donner une vue sur les objets métier. Il se concentre donc uniquement sur les objets important aux yeux des parties prenantes[2].

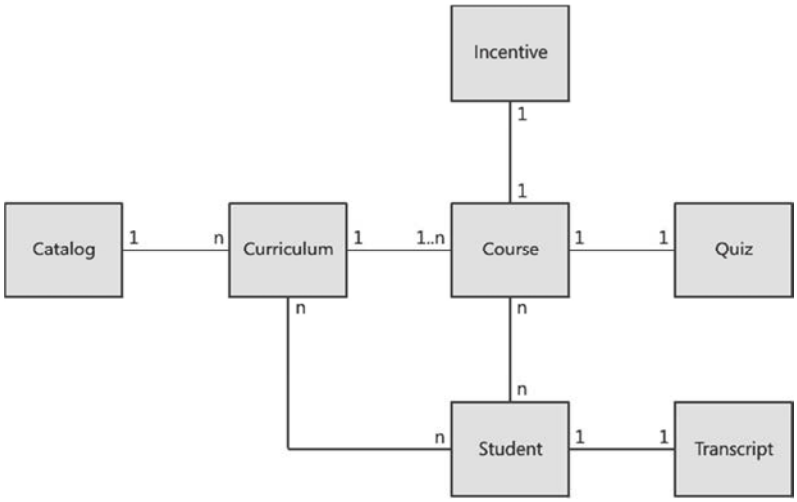

Figure VI-8 - Diagramme des données métier (Beatty, et al., 2012)

Ce diagramme possède des symboles concis, ce qui permet de représenter plus facilement de larges systèmes impliquant un grand nombre de données. En revanche, sa concision force l'analyste à réaliser un dictionnaire de données pour lequel il servira de fondations avec efficience.

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

Annexe III. Découpage du projet

UML : Diagramme de package (Package Diagram)

Un diagramme de packages est un modèle d'UML qui décrit l'organisation des éléments du projet et leurs interdépendances. Il rassemble ces éléments en groupes, chaque groupe étant nommé. Les éléments peuvent éventuellement se retrouver dans plusieurs groupes[1].

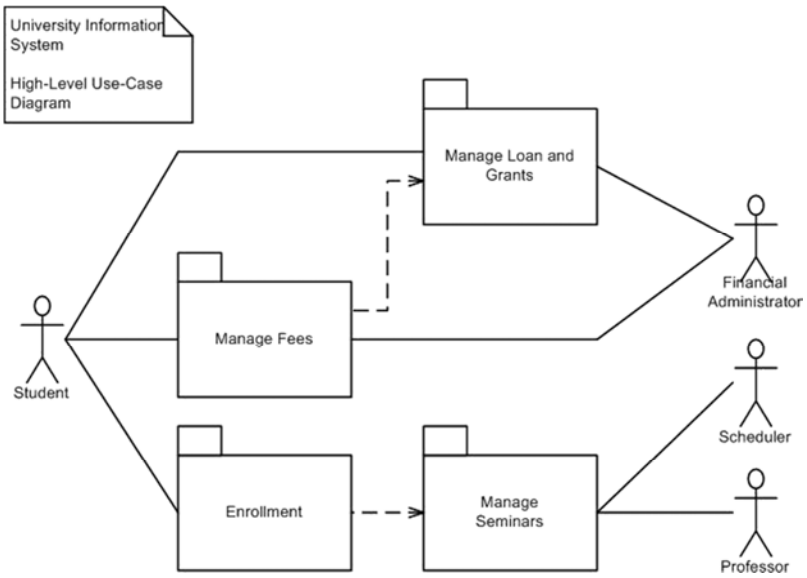

Figure VI-9 - Diagrame de package[2]

Le diagramme de package peut être utilisé en ingénierie des exigences pour modéliser la manière dont seront partagés les différents cas d'utilisation. On peut y représenter des dépendances et des hiérarchisations.

Structure de découpage du projet (Work Breakdown Structure)

Il s'agit d'une arborescence représentant la décomposition du projet en éléments de plus en plus petits. On peut ainsi découper le projet en itérations ou en livrables, et ces livrables en activités répondant ou correspondant aux différentes exigences[3].

[1] (Podeswa, 2009)
[2] Image tirée de http://www.agilemodeling.com/images/models/packageDiagramUseCases.jpg
[3] (International Institute of Business Analysis, 2009)

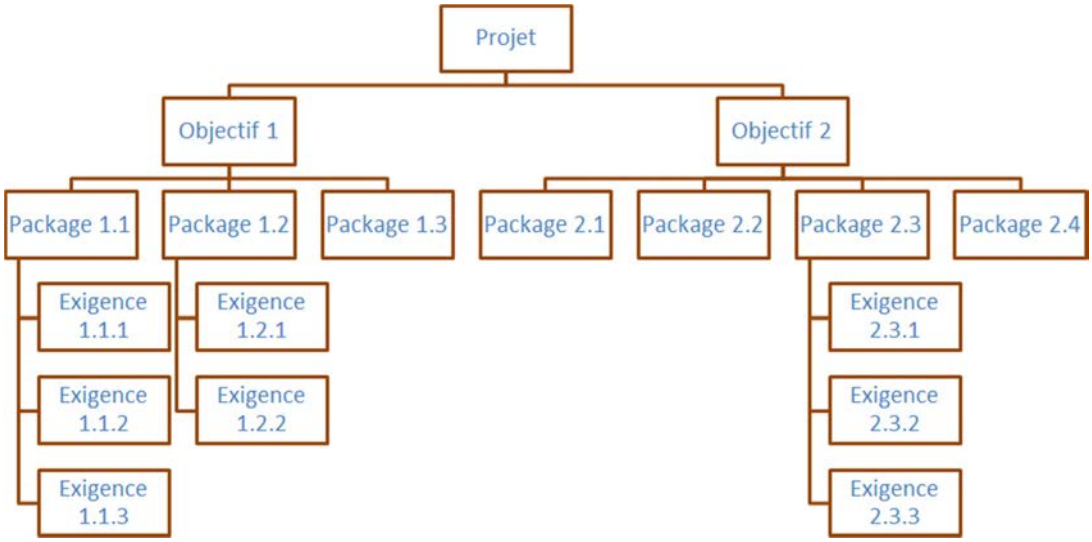

Figure VI-10 - Strucuture de découpage du projet

La décomposition se fait par exemple en package d'exigences depuis les objectifs du système comme présenté dans la Figure VI-10 D'autres décompositions existent en fonction de l'utilisation de la structure de découpage de projet.

RML : Arbre des fonctions (Feature Tree)

Ce modèle dispose les fonctions dans un arbre représentant l'entièreté de la portée du projet dans un seul schéma, dont la structure est inspirée des diagrammes d'Ishikawa[1]. La racine de l'arbre est constituée d'un nom court pour le produit et le standard conseille de ne pas descendre à plus de trois niveaux de détails.

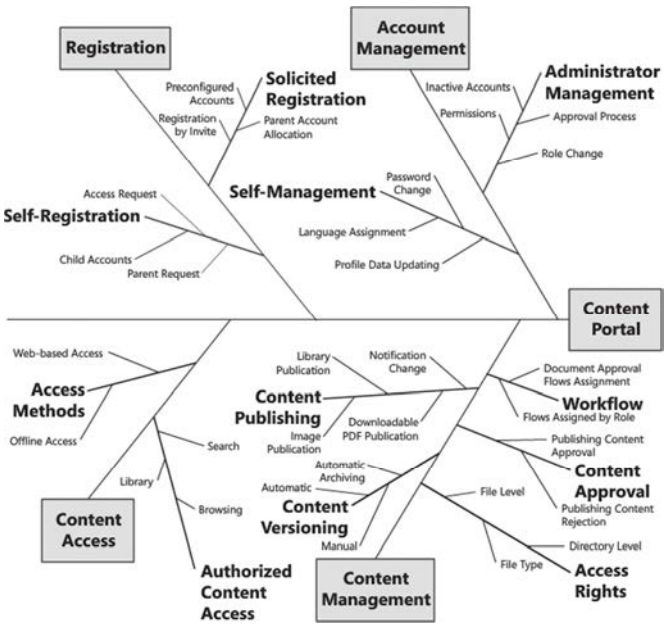

Figure VI-11 – Arbre des fonctions (Beatty, et al., 2012)

[1] (Beatty, et al., 2012)

Annexe IV. Enchaînement des états

UML : Diagramme d'états-transitions (State-Machine Diagram)

Le diagramme d'états-transitions permet de modéliser certaines exigences sur le comportement des objets-clés du métier et d'analyser leur cycle de vie. Les exigences qu'il permet de modéliser sont par exemple des réponses à des évènements, des conditions, des déclenchements de processus. Il offre aussi un bon moyen de s'assurer que les exigences fonctionnelles décrivent de manière complète et correcte les différents états et transitions requis pour l'objet[1].

Les différents symboles qu'il offre permettent de représenter, outre les contraintes citées ci-dessus, les états de l'objet étudié, les transitions entre ces états et les regroupements d'états dans des états composites.

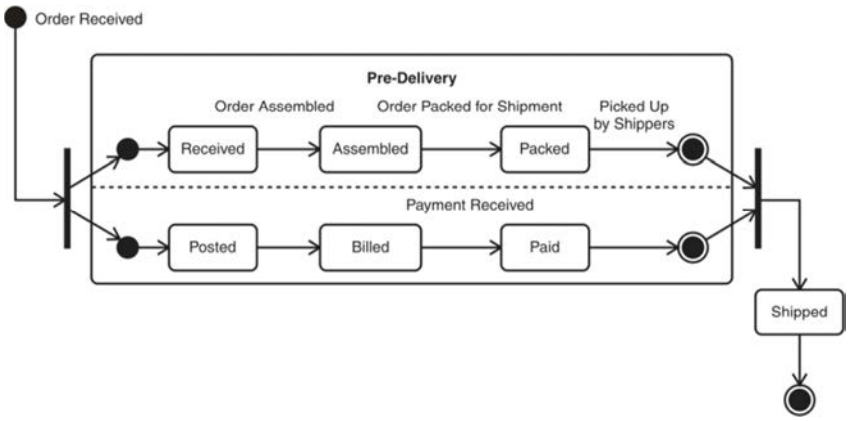

Figure VI-12 - Diagramme d'états-transitions (Podeswa, 2009)

Ce modèle donne à l'analyste un aperçu sur le cycle de vie d'un objet métier au travers des différents processus qu'il subit et des cas d'utilisation qui le font intervenir. Il permet également d'identifier les goulots d'étranglement, en collectant par exemple des données sur le temps passé dans un état. Enfin, il permet de vérifier qu'un jeu de tests assure la couverture complète des états de l'objet[2].

RML : Diagramme d'état (State Diagram)

Le diagramme d'état de RML possède un jeu de symboles plus limité que son cousin du standard UML. Cela peut cependant accélérer la lisibilité et augmenter l'espace disponible pour le schéma. Sa syntaxe permet de modéliser les états, les transitions, les états de départ et finaux.

[1] (Wiegers, 2003)
[2] (Podeswa, 2009)

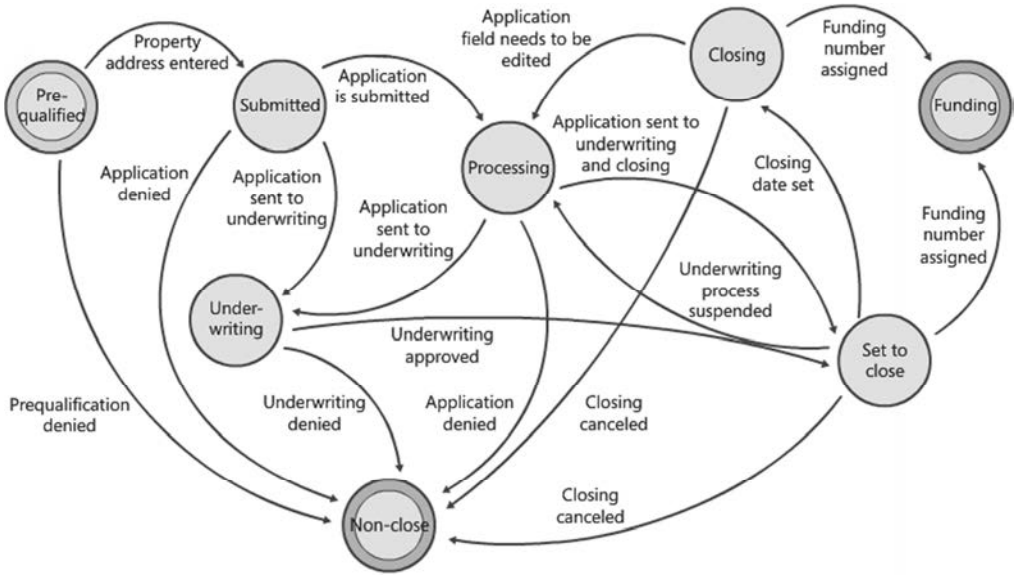

Figure VI-13 - Diagramme d'état (Beatty, et al., 2012)

Ensuite, des exigences fonctionnelles peuvent être découvertes si l'on analyse les conditions de déclenchement des transitions, les actions et transformations qu'elles initient, les données qu'elles génèrent, etc.[1]

Il permet de représenter les flux entre les états des objets métier clé, d'une manière plus facilement lisible que si la même information devrait être renseignée textuellement[2].

Il est facilement présentable aux parties prenantes pour vérifier la complétude du cycle de vie de l'objet, autrement dit vérifier que tous ses états ont été découverts. Enfin, il vient compléter d'autres modèles[3].

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)
[3] (Beatty, et al., 2012)

Annexe V. Flux

SA : Diagramme des flux de données (Data Flow Diagram)

Les diagrammes de flux de données représentent les différents processus clés du système, les acteurs, les centres de données et les échanges de données qui s'opèrent entre eux.

En représentant les entrées et les sorties des processus à certain niveau de détail, ils peuvent être utiles pour identifier l'ordre dans lequel ces données sont échangées. Ils s'avèrent utiles, dans un système comportant beaucoup de processus et utilisant un grand nombre d'objets métier, pour retracer le parcours de l'information[1].

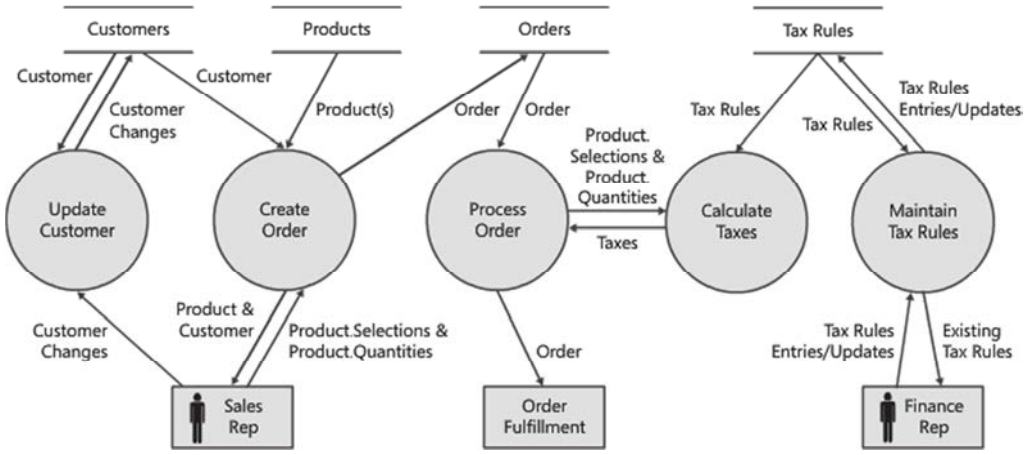

Figure VI-14 - Diagramme de Flux de Données (Beatty, et al., 2012)

Les diagrammes de flux de données sont à utiliser en conjonction avec d'autres modèles dans lesquels ils offriront de découvrir des manquements, telle qu'identifier les étapes manquantes d'un processus, les personnes manquantes dans un modèle des parties prenantes, ou encore, d'identifier des processus métier complètement oubliés[2].

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

UML : Diagramme de communication (Communication Diagram)

Le Diagramme de communication est orienté sur la structure d'un système, et sur l'étude des messages échangés entre les différents éléments de cette structure[1].

Le modèle pouvant être utilisé à différents niveaux, il sert à décrire l'agencement des unités de métier à un haut niveau, ou les sous-systèmes à un plus bas niveau. Il se concentre habituellement sur la modélisation de ce qui fait partie du système et non sur ses éléments adjacents. Ils peuvent être utilisés pour comprendre un système tel qu'il se trouve ou le modéliser tel qu'il est attendu.

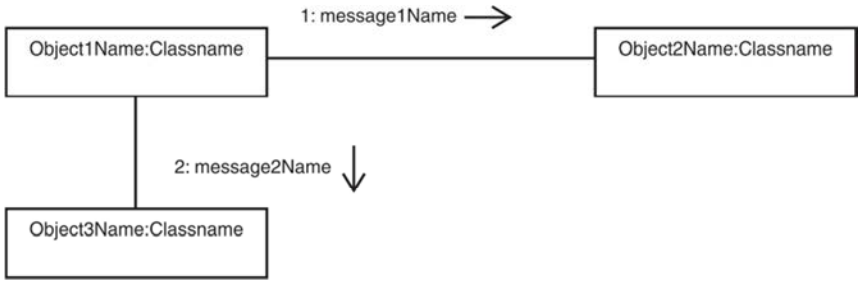

Figure VI-15 - Diagramme de communication (Podeswa, 2009)

[1] (Podeswa, 2009)

Annexe VI. Objectifs du système

UML : Diagramme des cas d'utilisation (Use cases Diagram)

Les cas d'utilisation sont les tâches que doit réaliser le système en réponse au besoin d'un utilisateur ou acteur. On peut donc parler pour cet acteur d'objectif attendu du système. Cet objectif représente une tâche à accomplir constituée d'interactions entre le système et l'acteur ayant initié la tâche et/ou d'autres acteurs[1].

Les acteurs peuvent être des utilisateurs du système ou des éléments de son contexte, tel qu'un autre système ou un composant.

Les objectifs ne sont pas détaillés dans le diagramme des cas d'utilisations, mais peuvent être dérivés (étendus) ou peuvent intégrer des sous-objectifs. Bien que d'aucuns considèrent les cas d'utilisation et leurs scénarios comme une intégralité indissociable, nous ferons de ces séquences d'interactions l'objet d'une autre modélisation, que nous décrivons en Annexe XI.

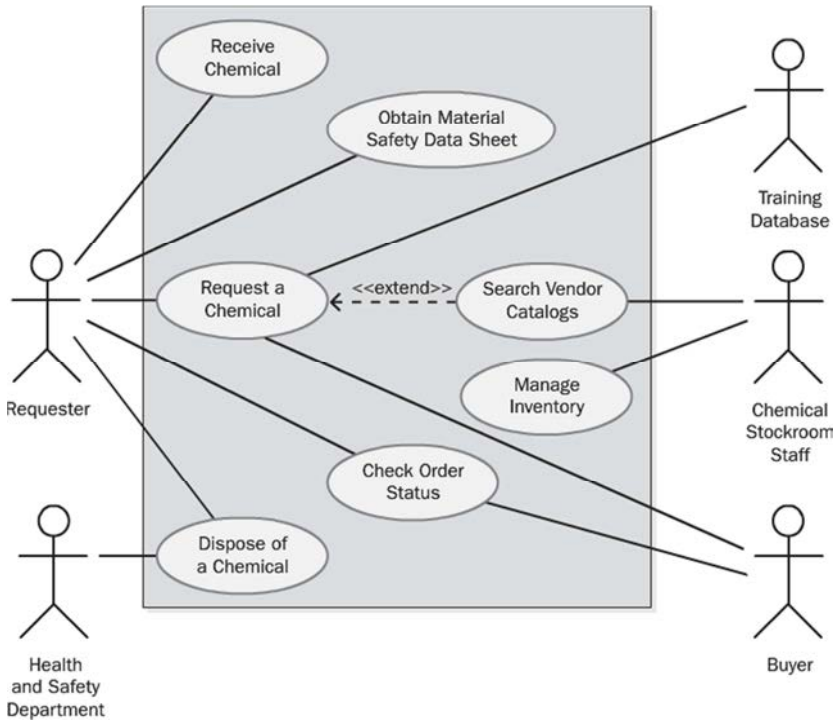

Figure VI-16 - Exemple de diagramme des cas d'utilisation (Wiegers, 2003)

[1] (Podeswa, 2009)

i* : Strategic Dependency

Les modèles i* partent du principe que les buts et les parties prenantes sont intimement liés. Ils augmentent l'idée (exprimée par exemple au moyen d'un diagramme des cas d'utilisation) de regrouper les objectifs par partie prenante, avec une notion de dépendance[1].

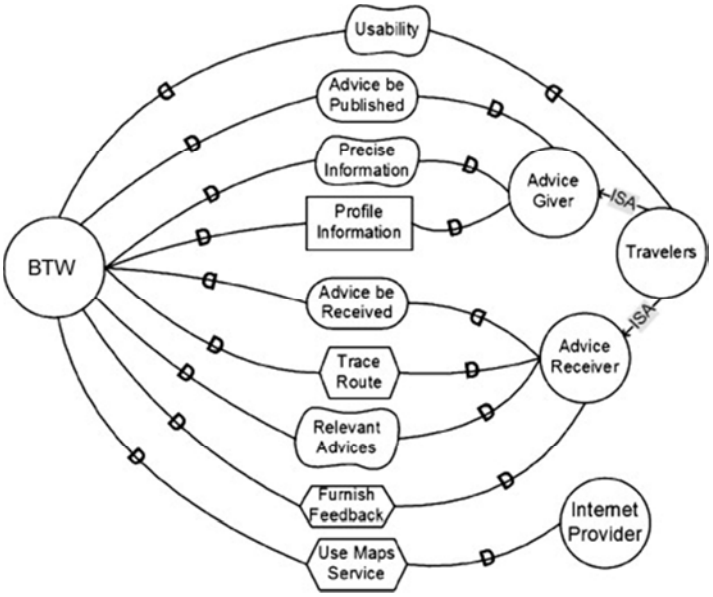

Figure VI-17 - i* Strategic Dependancy[2]

La notation *Strategic Dependency* permet de représenter les buts comme des relations de dépendance entre deux acteurs ou parties prenantes. Ces buts peuvent être un objectif (fonctionnel), une qualité (non-fonctionnel), une tâche, une ressource[3].

[1] (Alexander, et al., 2009)
[2] Image tirée de http://ars.els-cdn.com/content/image/1-s2.0-S0164121211001415-gr1.jpg
[3] (Petit, 2008-2009)

RML : Modèle d'objectifs métier (Business Objectives Model)

Ce standard a été imaginé pour s'assurer que l'équipe technique et les parties prenantes gardaient en ligne de mire la valeur devant être créée par le produit construit. L'avantage de pouvoir modéliser cette valeur ajoutée est celui de servir de base aux décisions à prendre quotidiennement sur les exigences. Ainsi, le modèle d'objectifs métier mettra en évidence que certaines fonctionnalités imaginées ne rentrent pas dans la création de cette valeur ajoutée aux yeux du métier[1].

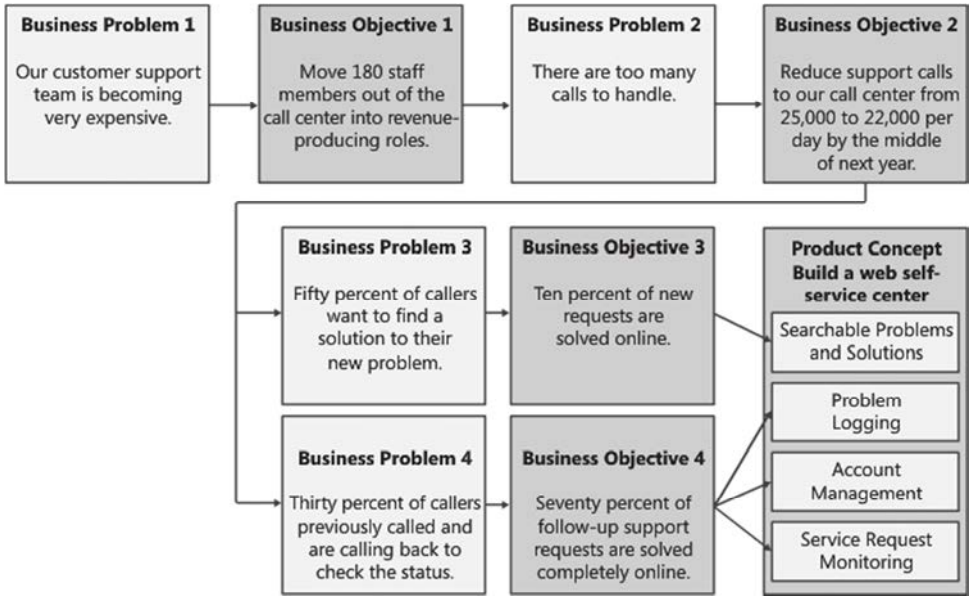

Figure VI-18 - Modèle d'objectifs métier (Beatty, et al., 2012)

Le modèle est agencé sous la forme d'un enchaînement d'éléments permettant d'arriver à des objectifs clé. Ces éléments sont des problèmes qui freinent le métier dans l'atteinte d'un but. Ils sont résolus par une « mesure de succès » ou « objectif métier » qui définit comment solutionner le problème. Ces solutions peuvent être suivis d'autres freins, ou, si le problème initial a été suffisamment étudié, mener directement à des fonctions clé du produit répondant à l'entièreté de la suite décomposée[2].

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

Annexe VII. Parties prenantes

Modèle en oignon (Onion Model)

Le modèle en oignon des parties prenantes (il s'agit de la carte des stakeholders, qui a été baptisée « *onion model* » par Ian Alexander[1]) est un diagramme qui représente la proximité des parties prenantes avec le projet, en les regroupant en couches, telles celles d'un oignon dont le centre est la solution attendue. Au plus les parties sont proches du centre, au plus celles-ci participent au processus.

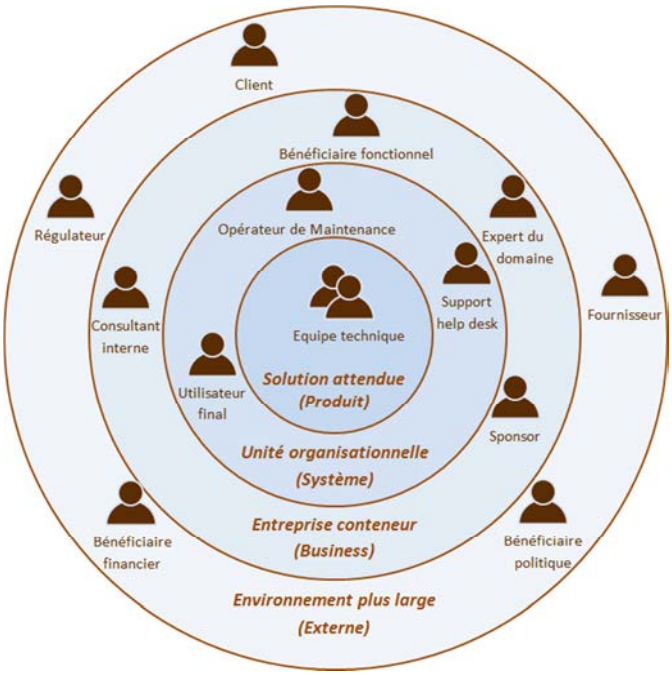

Figure VI-19 - Modèle en oignon des parties prenantes

La clarté de représentation rend ce modèle agréable à lire, à étudier et à présenter dans un workshop, sa sémantique étant facile à assimiler. De plus, il permet facilement d'identifier des rôles oubliés grâce à la représentation de rôles standards. L'analyste peut très facilement les identifier en posant aux personnes présentes des questions comme « Qui a ce rôle dans notre projet ? », « Qui interagit avec ce rôle ? », « Qui influence ce rôle ? »[2].

Ce diagramme présente le point faible de n'avoir qu'une seule mesure. Le principe étant de représenter de manière plus centrée les parties à plus forte implication dans la conception de la solution, le modèle ne dit pas grand-chose sur leur influence. Cependant, on peut noter que les personnes qui ne « toucheront pas à un clavier » durant la phase de développement sont souvent celles qui exercent le plus de pression sur le chef de projet[3]. Une version du modèle permet également de représenter l'influence des parties entre elles par un traçage de flèches sur le modèle.

[1] (Robertson, et al., 2006)
[2] (Alexander, et al., 2009)
[3] (Alexander, et al., 2009)

Matrice d'influence (Influence Matrix)

Une matrice d'influence permet de mettre en relation l'influence des parties prenantes avec leur impact, c'est-à-dire le niveau d'intérêt qu'ils portent au projet[1].

Bien que le Guide BABOK mette en avant cette relation, d'autres dimensions que l'impact et l'influence sont parfois mises en rapport via la matrice des stakeholders, à savoir la puissance (de faible à forte), le support (de négatif à positif), le besoin (de faible à fort)[2].

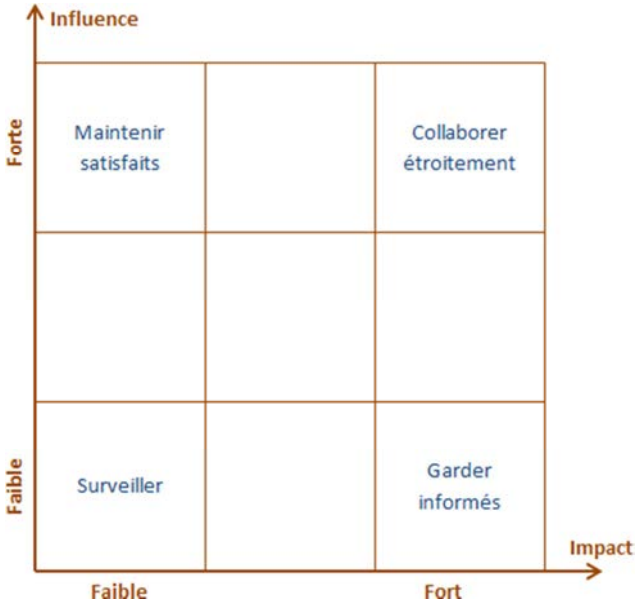

Figure VI-20 - Matrice d'influence des parties prenantes

L'enregistrement de ces indices dans l'outil d'IE permettrait facilement de visualiser, par leur placement dans la case correspondante, les personnes avec lesquelles il conviendra de collaborer. Cette méthode permet donc une réelle priorisation des demandes des parties prenantes.

Si l'outil permet d'afficher à tout moment le graphe à l'écran, ou de l'imprimer, on peut se fier également à cette méthode pour n'oublier personne, et l'on a une vue directe sur celles qu'il convient de contacter en fonction la tâche à réaliser.

RML : Diagramme de l'organisation (Organization Chart)

Ce diagramme dépeint la structure de l'entreprise en représentant de manière hiérarchique les membres la constituant, en les spécifiant de manière plus ou moins précise à l'aide de leur département, leur rôle et/ou leur nom. Le diagramme de l'organisation permet d'identifier et de classifier toutes les parties prenantes de l'organisation (utilisateurs ou membres influents) ayant un rapport avec les exigences à définir, modéliser ou valider[3].

[1] (International Institute of Business Analysis, 2009)
[2] (Thompson, 2011)
[3] (Beatty, et al., 2012)

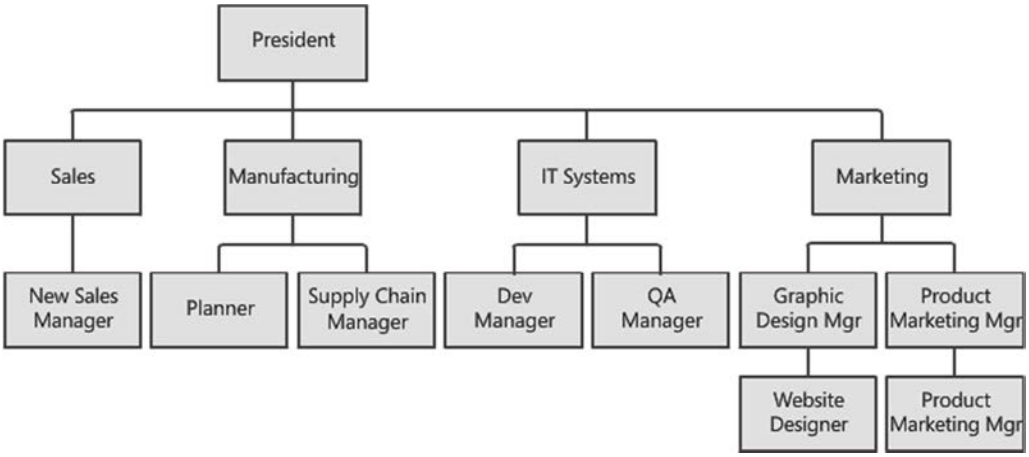

Figure VI-21 - Exemple de diagramme de l'organisation (Beatty, et al., 2012)

Le modèle aide à la détection des « possesseurs » des exigences, et de toutes les parties internes de l'entreprise. Il permet de savoir directement à qui s'adresser lorsqu'on doit approfondir des exigences impliquant un département ou rôle particulier. De plus, il permet de définir au début du cycle les termes et noms à utiliser au long du processus d'IE pour identifier les rôles et les personnes. Ces mêmes noms pourront servir par exemple aux entêtes des couloirs des diagrammes d'activités[1].

[1] (Beatty, et al., 2012)

Annexe VIII. Portée du produit

SA : Diagramme de contexte (Context Diagram)

Le diagramme de contexte permet de situer le produit au sein de son environnement. Le système y est représenté comme un tout, ainsi que les objets concrets ou abstraits qui l'entourent, et la ou les relations que le système a avec eux.

Certains diagrammes de contexte sont pourvus également d'une représentation des évènements[1] (signal déclencheur, paquet de données, entrée de contrôle, signal d'un senseur,…), par le biais d'un petit cercle à l'instar de celui se trouvant au bout du signal de la Figure VI-22.

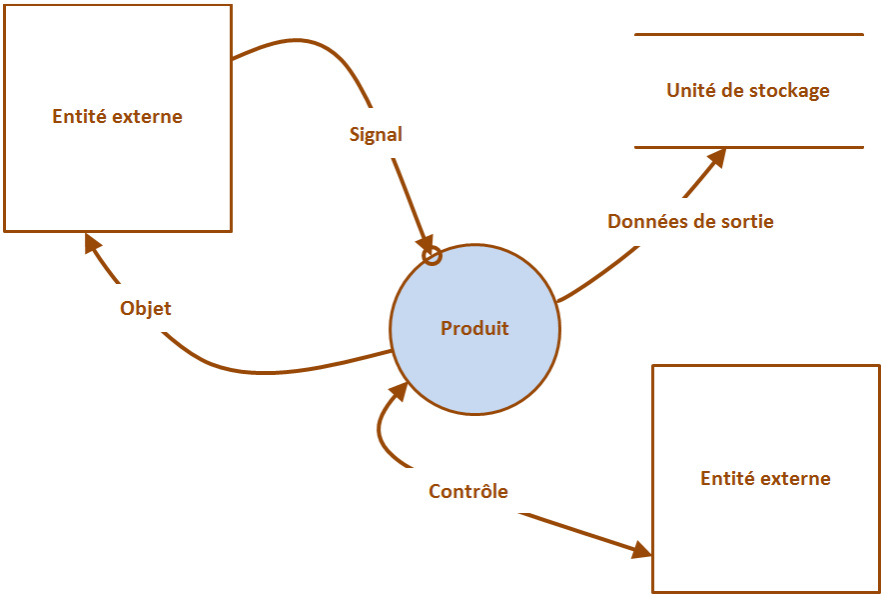

Figure VI-22 - Diagramme de contexte

Le diagramme de contexte, qui est une sous-classe d'un diagramme de flux de données (voir Annexe V)[2], permet déjà d'avoir un aperçu sur les interfaces à produire pour le produit, et par extension, sa portée. Le problème est de pouvoir définir la limite au bon endroit.

Graphe de décomposition fonctionnelle (Functional Decomposition Chart)

Ce type de représentation décompose, comme son nom l'indique, le produit en un arbre dont la racine est une unité du business (B.U.), ou « fonction », qui peut être déclinée en sous-fonctions. Les processus réalisés par l'unité sont ensuite représentés comme sous-nœuds, qui sont eux-mêmes décomposés en sous-processus, etc. Les processus non-décomposables sont appelés activités[3]. Il est une dérivation de la méthode de structure de découpage du projet (en anglais « *Work Breakdown Structure* », voir Annexe III).

[1] (Alexander, et al., 2009)
[2] (Pohl, et al., 2011)
[3] (Podeswa, 2009)

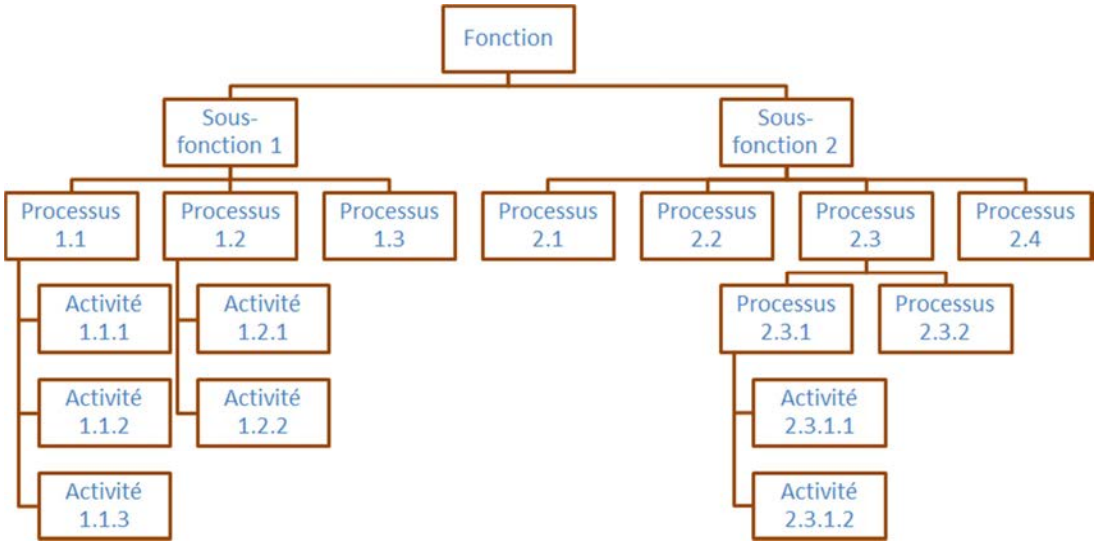

Figure VI-23 - Structure type d'un graphe de décomposition fonctionnelle

Le nom des nœuds et le nombre d'étages de l'arbre, ainsi que sa représentation, ne sont pas règlementés, le concept principal étant uniquement d'obtenir une décomposition hiérarchique numérotée des fonctions et processus du produit.

L'inconvénient de cette méthode est qu'elle ne représente pas le contexte du produit. Seule la partie interne du système est représentée. La limite est donc imaginaire, ce qui est représenté fait partie de la portée du produit, le reste n'en fait pas partie.

De plus, le diagramme n'offre pas de vue sur les interactions entres les B.U. et de ce fait, il n'y a aucun moyen de détecter qu'un composant a été oublié[1].

RML : Carte de l'écosystème (Ecosystem Map)

Une vue haut-niveau des interactions entre tous les éléments faisant partie du système et de son contexte peut être schématisée dans une carte de l'écosystème. Peuvent y être représentés les différents systèmes logiciels ou matériels, leurs interactions, les données majeures échangées, des groupements d'interactions, et la limite de la portée du projet. Enfin, dans de rares cas, le modèle autorise la représentation d'un acteur effectuant une connexion entre deux systèmes[2].

[1] (International Institute of Business Analysis, 2009)
[2] (Seilevel, 2011)

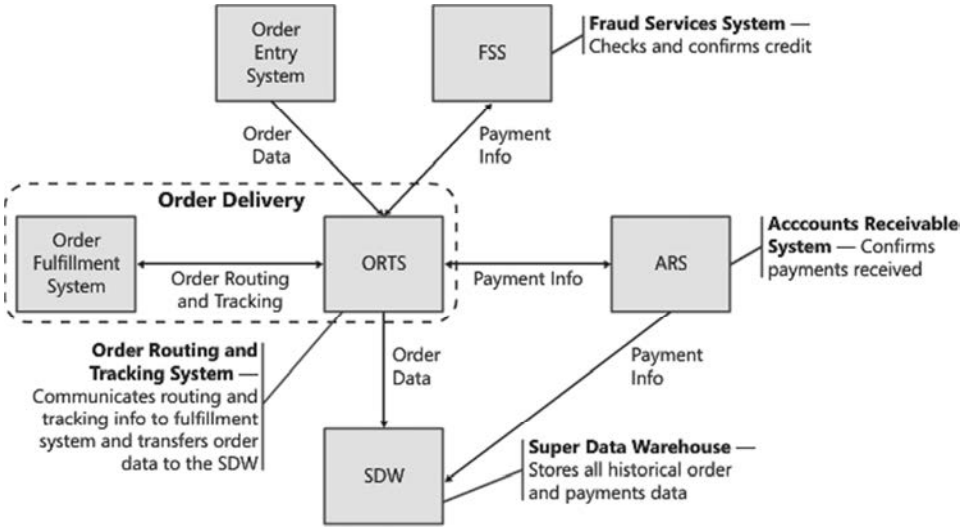

Figure VI-24 - Carte de l'écosystème (Beatty, et al., 2012)

L'avantage de ce modèle, s'il doit être mis en comparaison avec le diagramme de contexte, est qu'il permet non seulement de représenter les éléments se situant en dehors des limites du système, mais également ceux qui interagissent à l'intérieur de celui-ci[1]. Le système n'est donc pas représenté comme une « boîte noire » mais son fonctionnement est étudié, ce qui permet une meilleure interprétation des interfaces à définir, et une plus grande certitude qu'elles sont toutes capturées.

La représentation plus claire des interfaces (acteurs impliqués) permet également de dévoiler les exigences à éliciter à leur propos, ainsi qu'une meilleure visibilité de l'impact des changements qui seront apportés au système par la suite. Le modèle ne permet pas cependant une représentation précise des échanges, des séquences d'interactions, ni des éléments décisionnels[2].

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

Annexe IX. Priorité des exigences

Processus d'analyse hiérarchique (Analytic Hierarchy Process)

Il s'agit d'une approche par pondération sophistiquée qui se distingue par sa façon de déterminer les poids et critères. L'approche demande de trier les critères dans différents niveaux groupés en fonction de leur similitude[1]. Les niveaux doivent former une structure hiérarchique dans laquelle les exigences seront introduites, ce qui permet de décomposer un problème complexe en plus petits sous-problèmes.

Le cas typique est de décomposer le problème en trois niveaux[2], comme montré dans la Figure VI-25, cependant le processus d'analyse hiérarchique (pour laquelle nous utiliserons l'abréviation AHP de l'anglais) est prévu pour fonctionner avec un nombre de niveaux indéterminé.

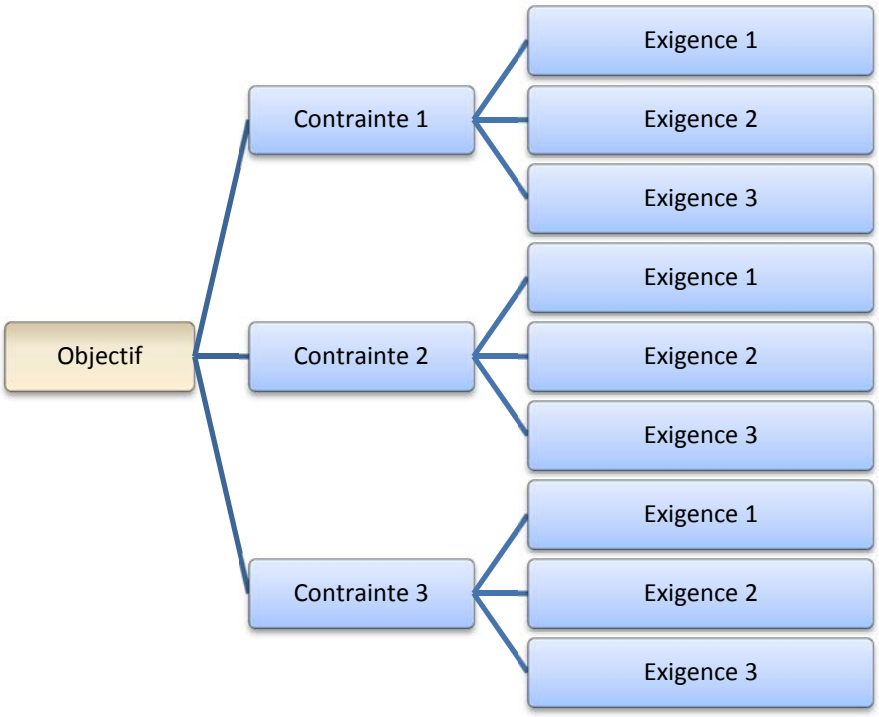

Figure VI-25 - Hiérarchie typique d'un processus AHP

Les éléments sont ensuite comparés deux à deux, en commençant d'abord par les contraintes (critères), puis par les exigences. Une combinaison binaire est opérée sur chaque niveau de l'arbre hiérarchique, par rapport aux éléments du niveau supérieur.

Cette méthode est pratique pour mesurer des critères non-quantifiables par l'attribution de scores ou poids relatifs et subjectifs, et est souvent utilisée également pour estimer les coûts et les valeurs des exigences[1].

[1] (Alexander, et al., 2009)
[2] (McCaffrey, 2005)

Un second avantage est de pouvoir garder une vue d'ensemble sur des problèmes complexes, d'abord en les découpant en plus petits sous-problèmes, ensuite en permettant d'obtenir une appréciation générale de chaque alternative, en enfin en permettant de considérer la priorité relative des critères[2].

Les éléments y étant comparés deux par deux, on peut se rendre compte qu'elle permet d'amener la totalité des données comparatives dans un état qui ne sera pas toujours consistant[3]. En effet le score attribué à une exigence A par rapport à une exigence B peut ne pas correspondre avec la différence entre les scores donnés à A par rapport à une exigence C, et celui donné à B par rapport à C.

$$\text{score}(\text{A}, \text{B}) \overset{?}{\leftrightarrow} \text{score}(\text{A}, \text{C}) - \text{score}(\text{B}, \text{C})$$

Cependant, la méthode est connue pour sa consistance justement, car elle offre un moyen d'estimer le degré d'inconsistance d'une matrice de scores et discute d'une limite raisonnable à ne pas dépasser[4]. Il est donc possible de garder une consistance logique des jugements utilisés pour déterminer les priorités.

Malgré l'adoption de cette technique dans le monde de l'ingénierie des exigences, peu d'outils d'IE l'implémentent actuellement. Lorsqu'elle est implémentée, cette méthode demande d'effectuer un grand nombre de comparaisons, ce qui peut vite devenir une lourde tâche lorsque le nombre de contraintes de comparaisons grandit[5].

Une étude[6] quantitative et qualitative montre également que la méthode n'est pas adaptée pour des projets où les parties prenantes sont fort distribuées géographiquement.

Matrice QFD

QFD est l'abréviation anglaise de « *Quality Function Deployment* » ou parfois appelé aussi « *Quality Function Development* », et désigne une méthode qui est surtout utilisée dans l'industrie au Japon[7]. La méthode met en relation les exigences des parties prenantes avec des réponses d'une part, et réponses entre elles d'autre part. La méthode utilise ainsi deux tableaux ; le premier est appelé la « Matrice QFD » et la seconde, « Maison de Qualité ». La mesure utilisée pour la comparaison donne un indice de qualité, représenté par des symboles.

Les réponses peuvent être soit des éléments de solution, soit des contraintes techniques, soit des caractéristiques de qualité.

---

[1] (Alexander, et al., 2009)
[2] (AIAC, 2010)
[3] (McCaffrey, 2005)
[4] (McCaffrey, 2005)
[5] (McCaffrey, 2005)
[6] (Ahmad, et al., 2010)
[7] (Alexander, et al., 2009)

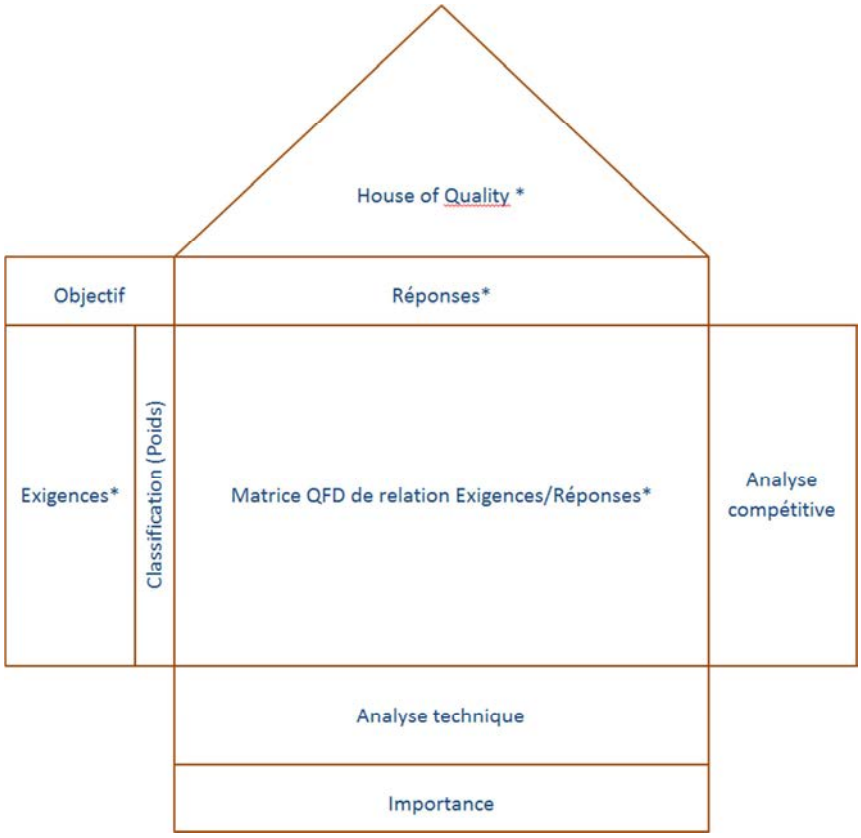

Figure VI-26 - Template de Matrice QFD

Comme le montre la Figure VI-26, les éléments de base du template de la matrice (éléments marqués d'un astérisque) peuvent être agrémentés de plusieurs éléments additionnels tels des scores d'importances et des sommes analytiques. Une grande quantité d'écoles existent, chacune ayant fait évoluer le modèle avec ses idées et ses besoins. Le but premier de la méthode, cependant, est d'apporter la plus grande satisfaction possible au client[1], c'est donc la « voix du client » qui viendra déterminer en grande partie les caractéristiques les plus importantes.

Le fait que la méthode conseille l'utilisation de symboles dans la matrice permet d'avoir une vision rapide des éléments positifs et des éléments négatifs. Ce procédé est reconnu pour sa psychologie compatible avec une certaine audience plus « commerciale ». Une force est de pouvoir modifier la cotation vers une notation numérique qui est mieux acceptée par un public plus mathématique désirant comprendre les algorithmes faisant partie du système[2].

Cette matrice peut être utilisée à plusieurs escients : assurance de qualité du produit, matrice de traçabilité, gestion des causes (« Rationale »)[3].

Un des problèmes inhérents à la matrice est qu'elle demande de connaître déjà un certain nombre de solutions[4]. Un second problème se pose lorsque les exigences évoluent ou lorsque d'autres

[1] (Wiegers, 2003)
[2] (Merts, 2009)
[3] (Davis, et al., 2001)
[4] (Wikipedia - Matrice QFD, 2012)

s'ajoutent : il faut refaire la quasi-totalité des calculs. Et malheureusement, les outils d'ingénierie des exigences, lorsqu'ils intègrent la matrice QFD, la fournissent essentiellement sous forme de template à imprimer et remplir manuellement. Il existe peu de logiciels proposant une Matrice QFD informatisée et en lien direct avec un outil de gestion des exigences : la plupart existent sous la forme d'un tableur Excel.

Calcul du retour sur investissement (ROI)

Il s'agit d'une approche « Coût-Valeur » qui consiste dans un premier temps à évaluer ou se mettre d'accord sur l'importance (« la valeur ») et le coût des exigences. Cela peut se faire par exemple à l'aide de méthodes telles que l'analyse AHP vue précédemment. Il convient ensuite de combiner ces deux facteurs dans un graphique ou une matrice pour avoir une meilleure visibilité de la priorité des exigences.

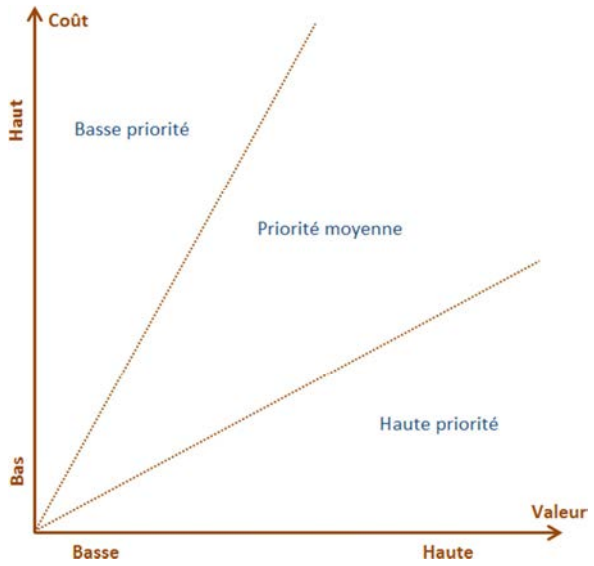

Figure VI-27 - Priorisation par combinaison de facteurs (« ROI »)

Le processus ROI (« *Return on Investment* ») est assez long car il nécessite d'établir les deux facteurs pour chaque exigence, mais le fait de les combiner donne une augmentation de la qualité de la priorisation[1]. Pour cette raison, ces facteurs peuvent être autres que ceux du ROI, par exemple[2] :

- Coût
- Valeur
- Temps nécessaire à l'implémentation
- Facilité technique d'implémentation
- Facilité organisationnelle d'implémentation
- Bénéfice pour l'entreprise

Il conviendra de déterminer les facteurs les plus pertinents pour le projet.

[1] (Alexander, et al., 2009)
[2] (Robertson, et al., 2006)

Annexe X. Processus métier

UML : Diagramme d'activité (Activity Diagram)

Grâce aux couloirs correspondant chacun a un acteur défini, le diagramme d'activité offre une représentation de la localisation des différentes tâches d'un processus métier. Outre les couloirs et les tâches, sa syntaxe définit des angles conditionnels (« *decision* ») ainsi que des points d'embranchement (« *fork* ») et de convergence (« *join* » et « *merge* ») qui aideront dans la description complète du séquencement du processus[1].

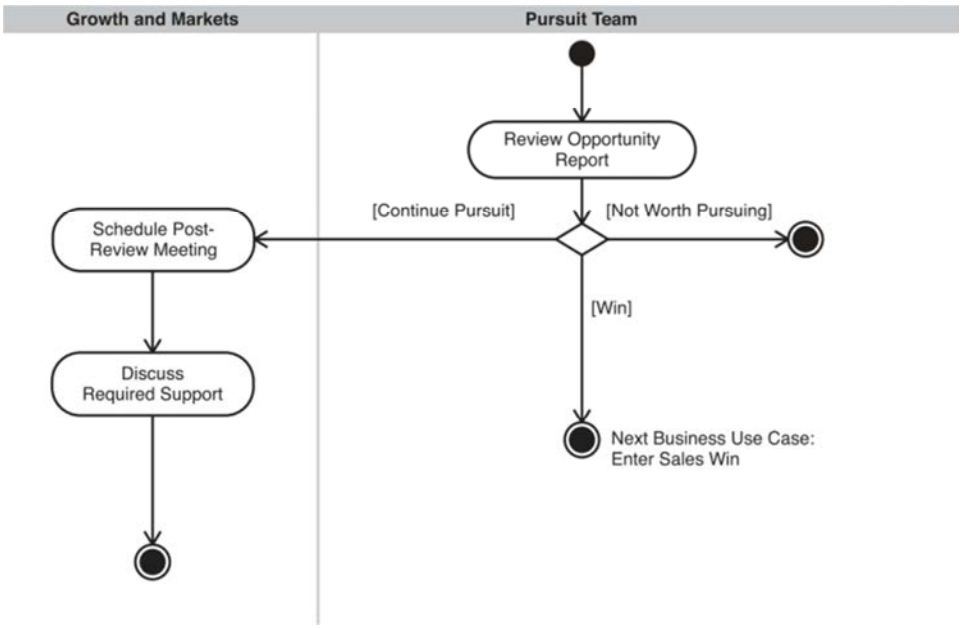

Figure VI-28 - Diagramme d'activité (Podeswa, 2009)

Un notation plus avancée prévoit également d'intégrer des objets qui sont utilisés ou créés à des étapes définies du processus.

BPMN : Diagramme de processus métier (Business Process Diagram)

Ce diagramme offre une sémantique très proche de celui du diagramme d'activité. Toutefois il apporte une richesse supplémentaire pour de la définition des évènements et niveau de précision plus élevé dans l'intégration des objets au sein du processus[2].

Les utilisations qui en sont faites sont généralement portées sur l'amélioration des processus métier. La nuance qu'il apporte en regard d'un diagramme d'activités est une meilleure définition des règles de séquencement[3].

[1] (Podeswa, 2009)
[2] (Podeswa, 2009)
[3] (Podeswa, 2009)

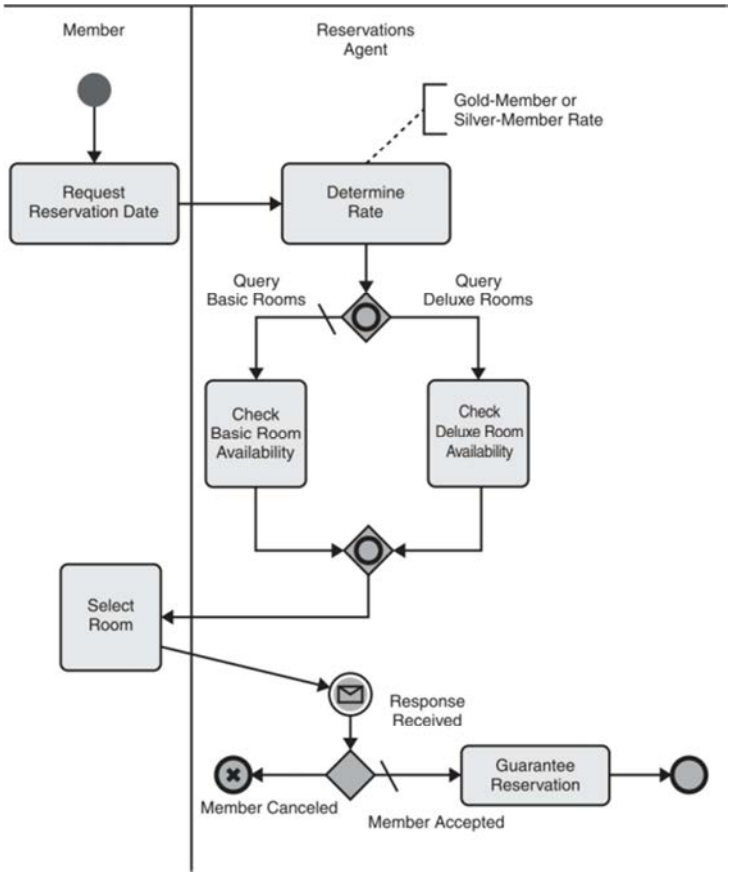

Figure VI-29 - Diagramme de processus métier (Podeswa, 2009)

RML : Flux de processus (Process Flow)

Variante supplémentaire des deux standards précités pour la définition des tâches effectuées par les utilisateurs au sein d'un processus métier, le flux de processus est caractérisé par une syntaxe plus simple et plus propice à la validation par les parties prenantes[1].

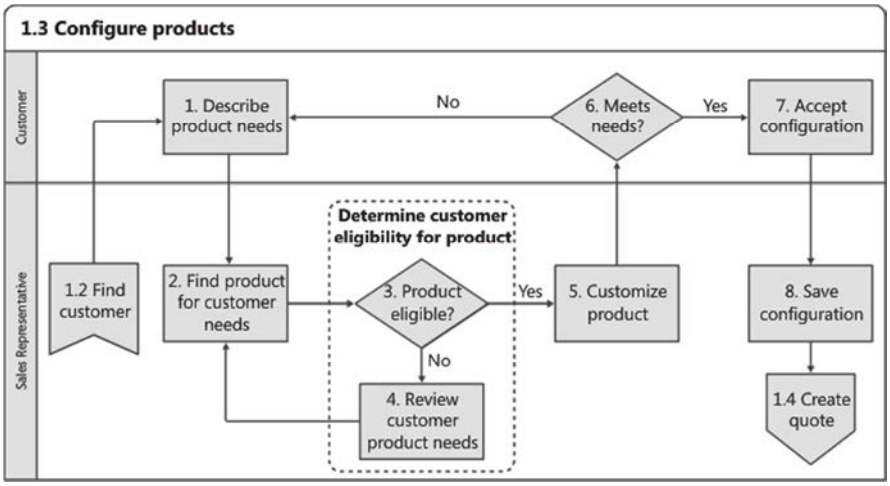

Figure VI-30 – Flux de processus (Beatty, et al., 2012)

[1] (Beatty, et al., 2012)

Annexe XI. Scénarios d'utilisation

Prototypage

Le prototypage consiste principalement en la rédaction de faux écrans de démonstration, non pour montrer à l'utilisateur à quoi le produit ressemblera, mais plutôt pour s'assurer que les fonctionnalités qu'il attend sont bien présentes et que le processus d'utilisation correspond bien à ses attentes.

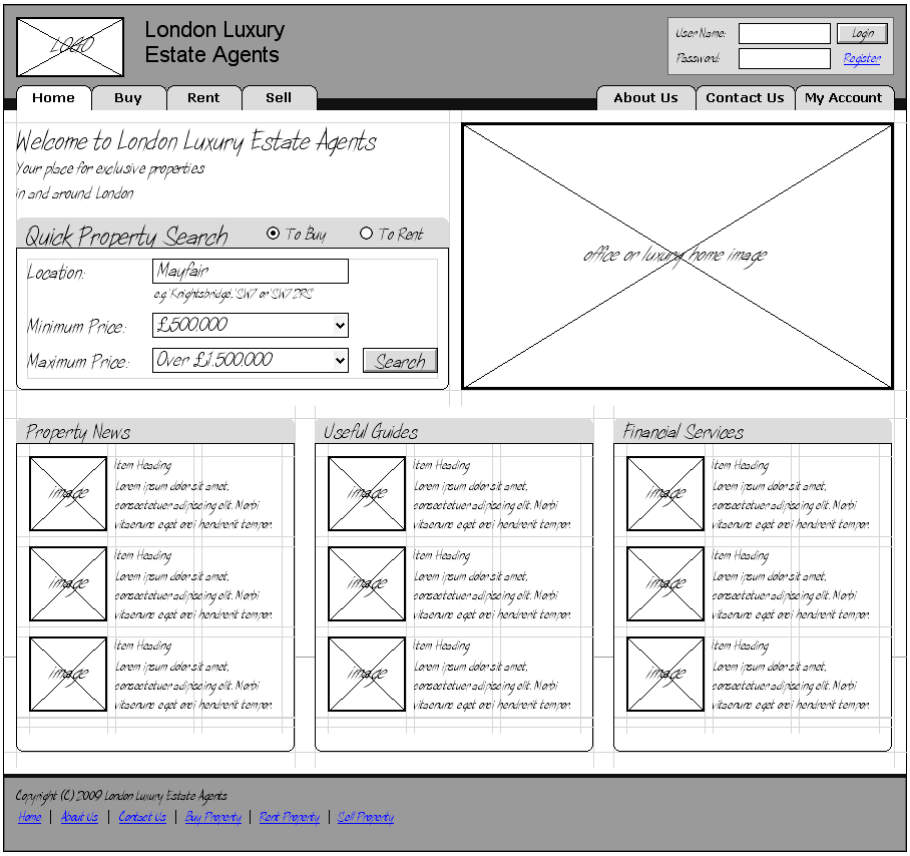

Figure VI-31 - Prototype d'un page Web[1]

[1] Image tirée du site http://www.carettasoftware.com/

UML : Diagramme d'activité (Activity Diagram)

Dépourvus de leurs éléments syntaxiques définissant les couloirs prévus pour représenter les acteurs, le diagramme d'activité convient également pour le détail d'un scénario d'utilisation[1]. En effet, les scénarios impliquent (sauf dans de rares cas) uniquement l'utilisateur ayant initié le processus.

Outre les étapes du scénario d'utilisation, des branchements y sont modélisables pour l'expression des cas alternatifs de celui-ci[2].

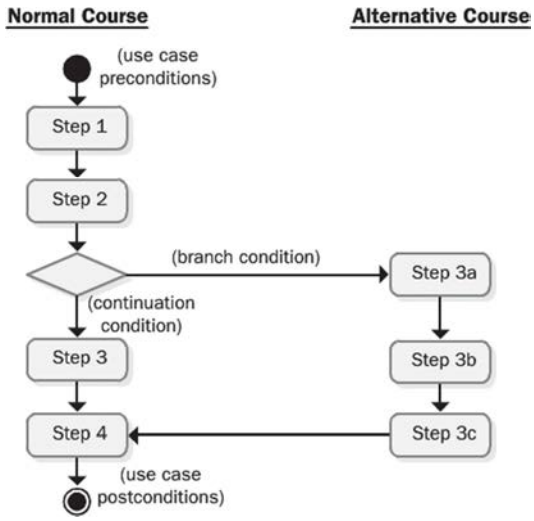

Figure VI-32 - Diagramme d'activité (Wiegers, 2003)

[1] (Pohl, et al., 2011)
[2] (Wiegers, 2003)

RML : Enchainement des interfaces utilisateur (UI Flow)

Ce modèle liste les différentes interfaces utilisateurs prévues pour le système. Il montre comment la navigation s'opérera entre ces différents écrans et permet également de représenter des zones réservées à certains utilisateurs[1].

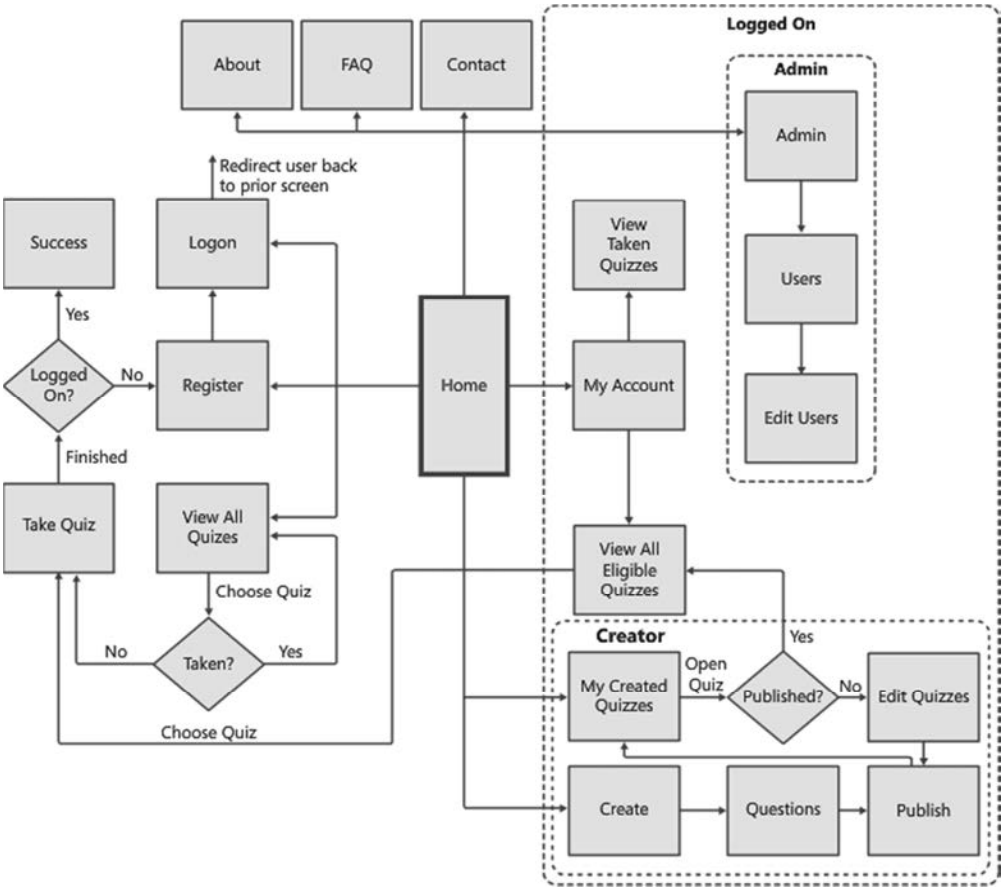

Figure VI-33 – Enchaînement des interfaces utilisateur (Beatty, et al., 2012)

Ses buts sont d'abord celui de la valider que la navigation permet d'atteindre tous les écrans, et ensuite d'améliorer l'a qualité d'utilisation du système (création de raccourcis par exemple)[2].

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

Annexe XII. Scénarios système

UML : Diagramme de séquence (Sequence Diagram)

Ce diagramme est optimisé pour la représentation des interactions entre plusieurs acteurs d'un système au sein d'un processus. Les acteurs peuvent être un utilisateur, une interface utilisateur ou une unité métier, et leur cycle de vie est représenté par la longueur de la ligne leur correspondant. Les interactions sont indiquées sous la forme de messages échangés ou d'actions effectuées, dont l'indication du séquencement est l'attrait principal du modèle[1].

La syntaxe est prévue pour pouvoir représenter à la fois un cas « normal » d'un scénario, et tous ses cas alternatifs[2]. Néanmoins, vouloir l'utiliser dans ce sens alourdit de façon conséquente le schéma lorsque le scénario comporte de nombreux embranchements.

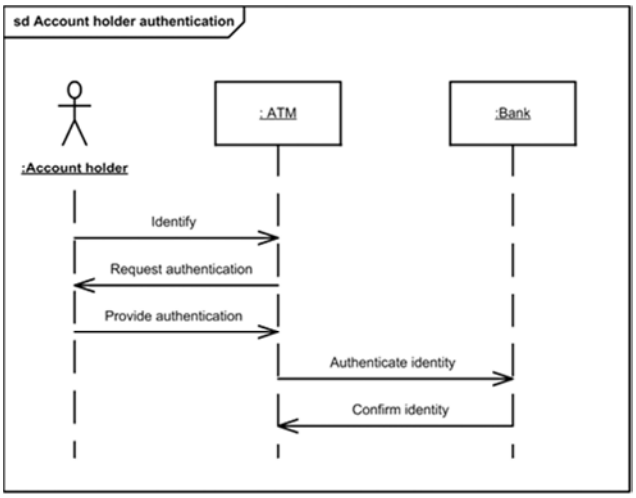

Figure VI-34 - Diagramme de séquence[3]

Ce modèle est peu adapté pour la validation du processus auprès des parties prenantes, à moins qu'ils soient familiers avec lui. En effet il réalise un grand pas vers la conception interne du système[4].

[1] (Podeswa, 2009)
[2] (Podeswa, 2009)
[3] Image tirée de http://www.jot.fm/issues/issue_2005_11/article5/images/figure2.gif
[4] (Podeswa, 2009)

RML : Flux de système (System Flow)

Le Flux de système est optimisé pour la représentation des étapes autonomes d'un processus du système. La syntaxe permet de distinguer la nature des étapes qui peuvent être soit une décision, une émission de signal, une réponse à un évènement, un référence à un sous-processus ou un processus externe, et enfin, une simple étape d'exécution[1].

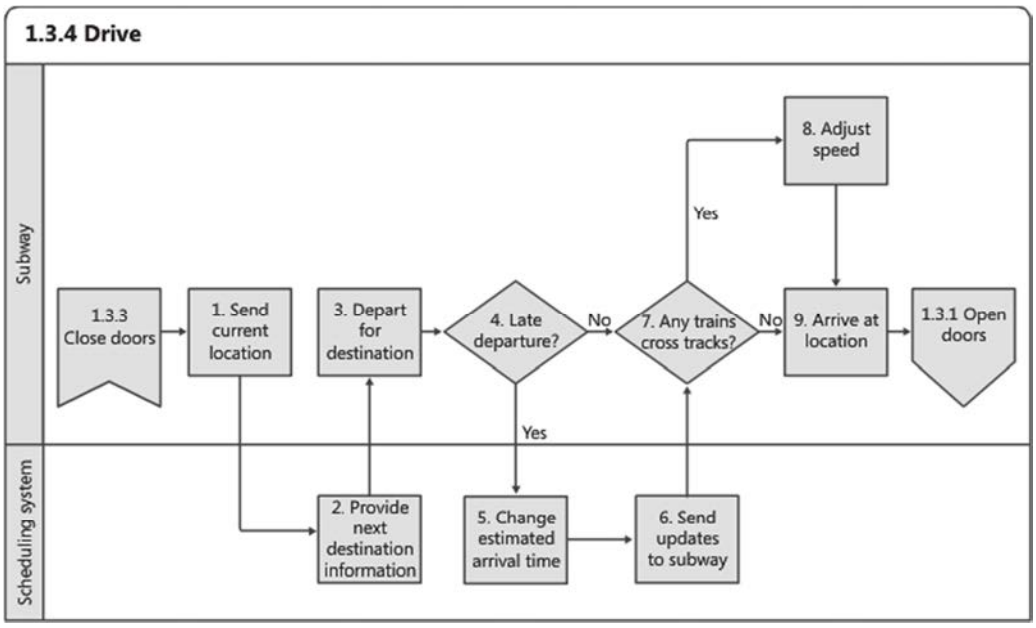

Figure VI-35 – Flux de système (Beatty, et al., 2012)

Ce modèle est un excellent outil lorsqu'il s'agit de faire comprendre aux parties prenantes les fonctionnements cachés des systèmes. Leurs interactions sont en effet peu évidentes à assimiler par un utilisateur qui ne les voit pas[2].

[1] (Beatty, et al., 2012)
[2] (Beatty, et al., 2012)

Annexe XIII. Traçabilité des exigences

Table de traçabilité

Une table de traçabilité permet d'obtenir une vue horizontale des dépendances d'une exigence. Sur chaque ligne est représentée une exigence, et à chaque colonne correspond une catégorie d'éléments telle que les cas d'utilisation, des exigences logicielles, la priorité définie, le risque évalué, un numéro de package, de cas de test ou de diagramme, de références à des fragments de code, etc. Les colonnes qui y sont présentes dépendent des liens de traçabilité qu'il a été jugé opportun de référencer dans l'outil d'IE[1].

User Requirement	Functional Requirement	Design Element	Code Module	Test Case
UC-28	catalog.query.sort	Class catalog	catalog.sort()	search.7 search.8
UC-29	catalog.query.import	Class catalog	catalog.import() catalog.validate()	search.12 search.13 search.14

Figure VI-36 - Matrice de traçabilité (Wiegers, 2003)

Le désavantage de cette méthode est qu'elle représente difficilement les liens existants entre différentes exigences de même niveau, à moins de devoir rajouter des colonnes qui pourraient vite alourdir la table. De même, si plusieurs éléments doivent être notés dans la même colonne pour une exigence, des problèmes de lisibilité et de lourdeur peuvent rapidement survenir.

Ensemble de matrices de traçabilité

Pour répondre aux deux problèmes de la table de traçabilité, une solution envisageable est de générer plusieurs matrices de traçabilité. « Table » et « Matrice » sont souvent considérés comme synonymes, nous considérerons cependant une différence : Les éléments des colonnes ne sont pas nécessairement de même type pour une table, alors que c'est le cas pour une matrice.

La méthode consiste donc un ensemble de matrices qui représentent chacune les liens entre les exigences et un autre ensemble de concepts, comme c'est le cas pour les cas d'utilisation dans la Figure VI-37.

[1] (Wiegers, 2003)

Functional Requirement	Use Case			
	UC-1	UC-2	UC-3	UC-4
FR-1	↵			
FR-2	↵			
FR-3			↵	
FR-4			↵	
FR-5		↵		↵
FR-6			↵	

Figure VI-37 - Matrice de traçabilité (Wiegers, 2003)

D'après Karl E. Wiegers, les liens de traçabilités intéressants sont les suivants[1] :

- Exigence système ⇔ Exigence logicielle
- Cas d'utilisation ⇔ Exigence fonctionnelle
- Exigence fonctionnelle ⇔ Exigence fonctionnelle
- Exigence fonctionnelle ⇔ Cas de test
- Exigence fonctionnelle ⇔ Elément d'architecture logicielle
- Exigence fonctionnelle ⇔ Autres éléments de conception
- Autres éléments de conception ⇔ Code
- Règle métier ⇔ Exigence fonctionnelle

[1] (Wiegers, 2003)

ANNEXES – RESSOURCES EXTERNES

Annexe XIV. INCOSE Requirements Management Tools Survey

Cette annexe reprend les contours de l'étude réalisée en août 2010 par l'INCOSE (*International Council on Systems Engineering*), à savoir les références des logiciels évalués ainsi que les critères d'évaluation. Nous ne reprenons pas ici les résultats qui peuvent être consultés en ligne à l'adresse http://www.incose.org/ProductsPubs/products/rmsurvey.aspx [mai 2012].

3.1. Outils concernés par l'enquête

1. Accept Requirements (Accept 360)

VERSION :	(non spécifié)
EDITEUR :	Accept Software
ADRESSE WEB :	http://www.acceptsoftware.com

2. Acclaro DFSS

VERSION :	5
EDITEUR :	Axiomatic Design Solutions, Inc.
ADRESSE WEB :	http://www.dfss-software.com

3. Aligned Elements

VERSION :	1.5
EDITEUR :	Aligned AG
ADRESSE WEB :	http://www.aligned.ch/

4. Avenqo PEP

VERSION :	1.2
EDITEUR :	Avenqo
ADRESSE WEB :	http://www.avenqo.com

5. Blueprint Requirements Center

VERSION :	2010
EDITEUR :	Blueprint Software Systems, Inc.
ADRESSE WEB :	http://www.blueprintsys.com

6. Cameo Requirements+

VERSION :	4.0
EDITEUR :	No Magic Inc.
ADRESSE WEB :	http://www.magicdraw.com

7. CASE Spec

VERSION :	8.15
EDITEUR :	Goda Software
ADRESSE WEB :	http://www.casespec.net/products.htm

8. Cognition Cockpit (Cockpit)

VERSION :	5.1
EDITEUR :	Cognition Corporation
ADRESSE WEB :	http://www.cognition.us

9. Contour by Jama Software (Contour)

VERSION :	2.9
EDITEUR :	Jama Software
ADRESSE WEB :	http://www.jamasoftware.com

10. CORE

VERSION :	7.0
EDITEUR :	Vitech Corporation
ADRESSE WEB :	http://www.vitechcorp.com

11. Cradle

VERSION :	6.4
EDITEUR :	3SL, Inc.
ADRESSE WEB :	http://www.threesl.com

12. Dimensions RM (DimRM)

VERSION :	10.1.4
EDITEUR :	Serena Software
ADRESSE WEB :	http://www.serena.com/products/rm/index.html

13. Enterprise Architect

VERSION :	7.1
EDITEUR :	Sparx Systems
ADRESSE WEB :	http://www.sparxsystems.com

14. Envision VIP

VERSION :	9
EDITEUR :	Future Tech Systems, Inc.
ADRESSE WEB :	http://www.future-tech.com/prod01.htm

15. IBM Rational DOORS

VERSION :	9.2
EDITEUR :	IBM
ADRESSE WEB :	http://www-01.ibm.com/software/awdtools/doors/productline/

16. IBM Rational RequisitePro

VERSION :	7.1
EDITEUR :	IBM
ADRESSE WEB :	http://www-01.ibm.com/software/awdtools/reqpro/

17. inteGREAT

VERSION :	4.7
EDITEUR :	eDev Technologies
ADRESSE WEB :	http://www.edevtech.com

18. IRQA

VERSION :	4
EDITEUR :	Visure Solutions
ADRESSE WEB :	http://www.visuresolutions.com/products/index.php

19. Kovair Global Lifecycle (Kovair)

VERSION :	5.5
EDITEUR :	Kovair Software, Inc.
ADRESSE WEB :	http://www.kovair.com

20. MagicDraw and SysML Plugin Version 16.5

VERSION :	16.5
EDITEUR :	No Magic Inc.
ADRESSE WEB :	http://www.magicdraw.com

21. MKS Integrity

VERSION :	2009
EDITEUR :	MKS Inc.
ADRESSE WEB :	http://www.mks.com

22. PACE

VERSION :	3
EDITEUR :	Viewset Corporation
ADRESSE WEB :	http://www.viewset.com

23. Polarion Requirements

VERSION :	2
EDITEUR :	Polarion Software
ADRESSE WEB :	http://www.polarion.com/products/requirements/index.php

24. Project & Test Engineering System (PTESY)

VERSION :	5.4
EDITEUR :	Andromeda s.r.l.
ADRESSE WEB :	http://www.andromeda-srl.com/PRODUCTS/PTESY/brochure.html

25. Psoda

VERSION :	5.02.1
EDITEUR :	e-LM.com Limited
ADRESSE WEB :	http://www.psoda.com

26. RaQuest

VERSION :	3.0
EDITEUR :	SparxSystems Japan Co., Ltd
ADRESSE WEB :	http://www.raquest.com/

27. ReqMan

VERSION :	2.0
EDITEUR :	RequirementOne Inc.
ADRESSE WEB :	http://www.requirementone.com

28. Reqtify

VERSION :	2010-1A
EDITEUR :	Geensoft
ADRESSE WEB :	http://www.reqtify.com

29. Requirements Manager (ReMa)

VERSION :	(non spécifié)
EDITEUR :	Accord Software and Systems Pvt. Ltd.
ADRESSE WEB :	http://www.rema-soft.com

30. RTIME

VERSION :	5.7
EDITEUR :	QAvantage
ADRESSE WEB :	http://www.QAvantage.com/

31. Teamcenter Requirements (Tc RM)

VERSION :	8
EDITEUR :	Siemens
ADRESSE WEB :	http://www.siemens.com/plm

32. TraceCloud

VERSION :	(non spécifié)
EDITEUR :	TraceCloud
ADRESSE WEB :	http://www.tracecloud.com

33. What To Do Next (WTDN)

VERSION :	(non spécifié)
EDITEUR :	4SQ Solutions LLC
ADRESSE WEB :	http://www.4sqsolutions.com/

34. workspace.com Requirements Management

VERSION :	(non spécifié)
EDITEUR :	workspace.com
ADRESSE WEB :	http://www.workspace.com/workspace/Requirements-Management-Software.html

3.2. Fonctionnalités visées par l'étude

La liste des fonctionnalités étudiées est reprise telle qu'énoncée par l'INCOSE, à savoir selon leur numérotation et dans la langue originale (anglais).

Les fonctionnalités sont réparties comme suit :

1. Capturing Requirements/identification

1.1. Input document enrichment/analysis: Using existing document information (such as glossary, index, etc.), aid the user in requirements analysis, identification of requirements, etc.

1.1.1. Input document change/comparison analysis: The ability to compare/contrast two different versions of a source document.

1.2. Automatic parsing of requirements: A mechanism for automatic identification of requirements by key words, structure, unique identifiers, etc. to create requirements from the text.

1.3. Interactive/semi-automatic requirement identification: The ability to identify requirements from a text file via interactive means such as mouse highlighting of the requirement text or prompting by the system "is this a requirement?"

1.4. Manual requirement identification: A manual means of identifying or creating requirements.

1.5. Batch mode operation: A mechanism for inputting/identifying requirements from outside of the tool.

1.5.1. Batch-mode document/source-link update: Does the tool have the ability to update existing linked documents from new/changed versions of the source documents without having to re-establish traceability links.

1.6. Requirement classification: Does the tool have the ability to classify/categorize requirements during identification

2. Capturing system element structure.

Once the requirements have been captured, the allocation of requirements to sub-system elements takes place. The tool must capture these elements so links/allocations can be made to those sub-systems elements.

2.1. Graphically capture systems structure: Can the tool graphically capture system implementation (such as architecture, functional decomposition, WBS, etc.) and display them graphically such that requirements can be linked to them.

2.2. Textural capture of systems structure: Can the tool textually capture system implementation (such as architecture, functional decomposition, WBS, etc.) and display them textually such that requirements can be linked to them.

3. Requirements flowdown.

Once the requirements have been captured and system architecture captured, requirements are allocated to the various system elements.

3.1. Requirements derivation (req. to req., req. to analysis/text): The ability to derive/create additional requirements and link between them such as requirement to requirement, or requirement to text (representing trade studies) to derived requirements.

3.2. Allocation of performance requirements to system elements (weight, risk, cost, etc.): The ability to link performance requirements to system elements such as weight, cost, throughput, etc. This also includes the ability to allocate portions of that performance requirement to system elements.

3.3. Bi-directional requirement linking to system elements: The linking of requirements to system elements can be accomplished from either end of the link--from the implementation back to the requirement or from the requirement down to the system element.

3.4. Capture of allocation rationale, accountability, test/validation, criticality, issues, etc.--if so how and what mechanism does it use?: Also critical, is the ability to attach rationale, assignments, criticality, test/validation and many other issues to the requirement, allocation, and the system element to which a requirement is linked.

4. Traceability analysis

Once the allocations are complete, the user will want the ability to see the links where they come from, where they go, and why they apply.

4.1. Identify inconsistencies (orphans... if so what kind of...): The tool should allow the user to identify inconsistencies such as unlinked requirements or system elements (orphans).

4.2. Visibility into existing links from source to implementation--i.e. follow the links: With the requirement links in place, the user needs the ability to follow the links to see where they come from and where they go to.

4.3. Verification of requirement (was it done, how was done): Throughout the life of the project, the requirement management tool will be used to verify that the requirements have been met. The tool should provide the ability to document that the requirement was fulfilled, how it was done, and who was responsible.

4.4. Requirement performance verification from system elements (roll up of actuals): Once performance requirements have been allocated to system elements, the requirements management tool should support the verification of those requirements by rolling up actuals and reporting on variances (this is the allocated weight versus the actual weight).

5. Configuration Management

5.1. History of requirement changes, who, what, when, where, why, how. : Once requirements have been captured, the requirement management tool should maintain a history of requirement changes, who changed it, when it was done, why it was done, etc. Some of this tracking could be automatic, others could be procedural such as a rationale for the change and how the change is to be accomplished.

5.2. Baseline/Version control: At various times the requirements will need to be baselined (saved and locked away). The requirements management tool should support this along with the ability to compare and contrast between various baselines.

5.3. Access control (modification, viewing, etc.): The requirements should be able to be protected from modification, viewing, etc. by individuals or groups.

6. Documents and other output media

6.1. Standard specification output (if so what kind): The requirements management tool should output documentation in various military/commercial standard formats (490, 2167, etc.).

6.2. Quality and consistency checking (spell, data dictionary): The tool should also support document quality and consistency checking through spell checking, data dictionaries, acronym tables, etc.

6.3. Presentation output: Once the information is loaded, the requirements management tool should support the generation of presentation quality charts and graphs.

6.4. Custom output features and markings (user definable tables, figures, security markings..): The tool should support the output of documents in finished form including page security markings, graphics/figures, user definable tables, indexes, etc.

6.5. WYSIWYG previewing of finished output: The tool should allow the user to view the document on-screen in finished format.

6.6. Status reporting: Tool users need to status information in the requirements management tool.

6.6.1. Technical Performance Measurement status accounting: Status current technical performance of various allocated performance requirements and monitor progress towards goals.

6.6.2. Requirement progress/status reporting: Status reporting on current compliance/non-compliance to various requirements

6.6.3. Other ad hoc queries and searches: The requirements management tool should support ad hoc queries and searches per the user's discretion.

7. Groupware.

Since Systems Engineers rarely work as individuals, the ability for a team of engineers to look/work on the same information at the same time is critical.

7.1. Support of concurrent review, markup, and comment: The tool should support a team of engineers reviewing, marking up, and commenting on requirements or implementation alternatives.

7.2. Multi-level assignment/access control: Access by the team to the database must be tempered by multi-level access control (i.e. the ability to protect things from being modified). This also includes the ability to submit changes into an approval cycle (for acceptance/voting) before committing the changes to the tool for everyone to see.

8. Interfaces to Other Tools

8.1. Inter-tool communications: Requirements management must have the ability to communicate requirements to other domain-specific design tools (CASE, EE, etc.).

8.1.1. Interfaces to other tools: What tools will your requirements management tool interface with or talk to?

8.1.2. External Applications Program Interface available: To support the wide variety of tools in use by engineers, the requirements management tool should have programmable access to the information contained in the tool's database (to get access to and deposit information).

8.1.3. Support Open database system (standard query access): Does the tool support Open Database standards such as standard query languages or exchange formats?

8.1.4. Import of existing data from various standard file formats? Does the tool have the ability to import existing data (such as a ASCII text file containing link information) to create structures within the tool without having to re-enter the information?

8.1.5 Support for Data Exchange Standards (AP-233, XML...)

8.2. Intra-tool communications

8.2.1. Exchange of information between same tool, but different installations: Since the tool will be used at different sites and different projects, how does the tool exchange information between different tool installations or databases?

8.2.2. Consistency/comparison checking between same-tool datasets: Does the tool support comparing/contrasting of different same-tool datasets to allow consistency and verification checking?

9. System Environment

9.1. Single user/multiple concurrent users: Is the tool support a single user or multiple concurrent users?

9.2. Multiple Platforms/Operating Systems? Which platforms and operating systems does the tool run on?

9.3. Commercial vs. unique database: Does the tool use a proprietary or commercially available database?

9.4. Resource requirements: Please identify hardware/software configuration requirements:

9.4.1. Memory requirements

9.4.2. CPU requirements

9.4.3. Disk space requirements

10. User Interfaces

10.1. Doing one thing while you are looking at another: Does the user have the ability run a report and look at a requirement at the same time?

10.2. Simultaneous update of open views: If the tool allows for multiple windows/views into the tool--does a change in one view automatically reflect in all other views?

10.3. Interactive graphical input/control of data: Does the tool support graphical input and manipulation of data?

10.4. Which window's standard do you follow? If your tool supports a window's standard, which one(s)?

10.5. Executable via scripts (recordable) or macros: Does the tool allow the user to create and playback commands or macros that allow the user to automate various tedious tasks?

10.6 Web browser interface?

10.7 Edit Undo Function Support

11. Standards

11.1. Which military/commercial standards does your tool comply with--including database standards, output document standards, exchange standards, display/graphics standards, etc.?

12. Support and Maintenance

12.1. Warranty: Does your tool have a warranty, if so what is it?

12.2. Network license policy: Does the tool support network licensing (floating, node locked, etc.), if so which license manager?

12.3. Maintenance and upgrade policy: How often are software updates released; are updates separately priced items, etc.?

12.4. On-line help: Are the user's manuals on-line; is there on-line help with the tool?

12.5. Provide Internet Web page location for company/tool info

12.6. Phone support: What type of phone support is available from the tool supplier?

12.7 Support User's Group

13. Training

13.1 Tool specific training classes

13.2 Training available at customer's location

13.3 Recommended training time: What is the recommended training time for a user to become proficient in using the tool?

13.4 Software installation with only basic training

14. Additional Comments

14.1 What other requirements management features do you as a tool supplier think are important (modeling, etc.)?

Annexe XV. RE-Tool Evaluation Approach

Cette annexe reprend les descriptions complètes des frameworks présentés par R. Matulevicius dans sa thèse « *Process Support for Requirements Engineering* ».

3.1. Framework for Functional RE-tool Requirements

Representation dimension

Le détail des activités du framework se trouve dans la Figure VI-38.

Agreement Dimension

Le détail des activités du framework se trouve dans la Figure VI-39.

Specification Dimension

Le détail des activités du framework se trouve dans la Figure VI-40.

	Features	Activities How does the RE-tool…	Activity description
Representation dimension	**FEF1.1.** Specify uniquely identifiable description using informal language.	**FEF 1.1.1** provide natural language description.	*Natural language* is the language in which humans communicate in everyday life. The requirements could also be defined using *professional language* (Pohl, 1994, Krogstie and Sølvberg, 1996), which is used by a set of persons working in a certain kind of area.
		FEF 1.1.2 allow specifying unique identification (ID) for each separate requirement.	Requirements ID is assigned when requirement is created and entered to a requirements database. It should be unique for each individual requirement.
		FEF 1.1.3 allow importing of requirements and their description from text document.	The initial user needs are usually stored in requirements documents that are text files. The tool should have means to import this information to the tool database, using semi or fully automated functionality.
	FEF1.2. Specify requirements using semi-formal language(s).	**FEF 1.2.1** provide tools for semiformal language description.	*Semiformal language* (Pohl, 1994) is language with a precisely defined vocabulary and syntax, but without a precisely defined semantics (e.g., ER-diagrams, OMT, UML, and DFD). The particular languages maintained in the RE-tool, are specified using non-functional process requirements.
		FEF 1.2.2 provide forward/ backward traceability between semiformal, and informal, formal descriptions.	While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceability relationships between different representation forms.
	FEF 1.3. Specify requirements using formal language(s).	**FEF 1.3.1** provide tools for formal language description.	*Formal language* (Pohl, 1994) is a language with a precisely defined vocabulary, syntax and semantics (e.g., Z-schemas, beta-notations, and aggregation models). The particular languages maintained in the RE-tool, are specified using non-functional process requirements.
		FEF 1.3.2 provide forward/ backward traceability between formal and informal, semiformal descriptions.	While having requirements definition in different representation languages, it is important to keep requirements uniqueness and maintain traceability relationships between different representation forms.
	FEF 1.4. Define traceable associations between requirements and the different elements of requirements specification.	**FEF 1.4.1** provide functions for testing traceability between informal, semiformal and formal requirement description.	By defining traceability between different representation forms, it is important to ensure that the requirements model is not changed using different views. The activity is partially duplicated with FEF1.2.2 and FEF1.3.2. However it is important to have it here as the RE-tool evaluation purposes could differ in different environments and previous activities would not be chosen for assessment.
		FEF 1.4.2 create parent-child traceable relations between requirements.	The activity defines hierarchical parent-child traceability. Parent here is understood like a higher abstraction level requirement, which consists of lower abstraction level requirements – children.
		FEF 1.4.3 maintain peer-to-peer traceable relations between requirements.	Peer-to-peer relationship defines requirements trace on the same hierarchical level (on the same abstraction level).
		FEF 1.4.4 maintain traceable relation between various related information.	Related information could be characterised as additional requirements views, figures, requirements documents, elicitation, or negotiation material.
		FEF 1.4.5 maintain forward/ backward traceability between a source of requirements, the requirements and design.	Requirements source is understood as the material (document, related system or person), from where (or whom) requirements are elicited. Design here is the system description, which specifies how the system should be built and implemented.
	FEF 1.5. Connect seamlessly with other tools and systems, by supporting interoperable protocols and standards.	**FEF 1.5.1** allow importing/exporting requirements description from/to text documents.	Importing of textual information from text and/ore graphical documents eases requirements model construction. Exporting of textual and/or graphical requirements information helps to analyse them using other software tools. Both import and export functionality could be fully or semi-automated. The activity includes analysis of the interoperability relationships and associations between the RE-tool and other tools used in an organisation.
		FEF 1.5.2 allow importing/exporting requirements description from/to graphical documents.	

Figure VI-38 - Representation Dimension

	Features	Activities How does the RE-tool…	Activity description
Agreement dimension	**FEF 2.1.** Maintain an audit trail of changes, archive baseline versions; and engage a mechanism to authenticate and approve change requests.	**FEF 2.1.1** maintain user authentication to the system (i.e. user name, password).	RE-tool users are stakeholders of the RE process. When speaking about the product produced with the RE-tool, users consist of different groups: those who are responsible for product development, introduction and maintenance; those with financial interest, responsible for the sale or purchase; and those who have an interest in product use.
		FEF 2.1.2 allow grouping users into different groups.	The RE-tool has to allow definition of user groups, and assignment the users to them.
		FEF 2.1.3 allow creating different views (according to documents, requirements, attributes) for different groups of stakeholders.	Different user groups could have different interest in requirements activities. The tool should allow display of various requirements views and tool functionality according to the user group or individual user needs.
		FEF 2.1.4 register agreement/ rationale/ discussion/ negotiation/ changes/ history of requirements and by how it was achieved.	*Rationale* is a reference to a set of information which provides an explanation why the requirement has been included. The negotiation, discussion and history maintenance are the means to create and to maintain the requirements rationale.
		FEF 2.1.5 call the earlier requirement description/ versions and register them into history context.	*History* is chronologically ordered states of individual requirements or requirements model. The RE-tool has to support history control mechanism and keep change track. The earlier requirements model (or requirement) versions should be possible to upload.
	FEF 2.2. Classify requirements into logical user-defined groupings.	**FEF 2.2.1** allow specifying attributes/ properties of the requirement.	Definition of requirements attributes and properties include requirements prioritisation activities, where different requirements or their groups are characterised according to the importance, stability, time to market and similar characteristics.
		FEF 2.2.2 provide sorting according to different attributes/ properties.	Requirements sorting and filtering according to defined attributes and properties, helps to narrow analysis scope and to find the relevant requirements to a user.
		FEF 2.2.3 provide filtering according to different attributes/ properties.	
	FEF 2.3. Support secure, concurrent cooperative work between members of a multidisciplinary team, which may be geographically distributed.	**FEF 2.3.1** provide platform independent interface for geographically distributed users.	Platform independent interface suggests means for cooperative geographically distributed teams to access the requirements model through Web browsers.
		FEF 2.3.2 allow making a copy for modification of an already approved version of requirements description in different abstract levels (document, requirement).	Modification of requirements model versions leads to working on the same requirements data at the same time. Approval of requirements helps to reach agreement about the requirements model. The requirements model is the object of negotiation and agreement about the requirements, and their changes. The RE-tool should ensure that different versions of the requirements model would not overlap but be artefacts to reach the agreement.
		FEF 2.3.3 provide a change approval cycle for multiple change negotiation and approval before posting into common repository.	
	FEF 2.4. Maintain a comprehensive data dictionary of all project components and requirements in a shared repository.	**FEF 2.4.1** provide the single repository or data and concept dictionary.	Data dictionary provides definitions of the main elements of the requirements model, RE-tool functionality, and related information.
		FEF 2.4.2 provide separate data dictionaries for non-technical users and technical users.	The RE-tool is used by users who have knowledge about the RE and software development process; and users who do not have such a knowledge. The RE-tool has to distinguish between both groups of users providing appropriate data dictionaries or suggesting process scenarios.
		FEF 2.4.3 provide the help system to the users.	The RE-tool has to provide help functionality, where all the RE-tool functions and features would be defined and explained.

Figure VI-39 - Agreement dimension

	Features	Activities How does the RE-tool…	Activity description
Specification dimension	**FEF 3.1.** Collect and store a common system's and a product family's domain requirements.	**FEF 3.1.1** enable selection and extraction of common domain requirements and requirements which differentiate systems in product line.	The activity emphasises the requirements reuse from the requirements repository, which is maintained by the RE-tool. Requirements from the repository could be selected using, for instance, free-based or/and discriminant-based methods (Mannion *et al.,* 1999; Kaindl and Mannion, 2005)
		FEF 3.1.2 incorporate requirements to a concrete project.	Incorporation of requirements (and their groups) into appropriate places of the requirements model helps to create the requirements specification faster as these requirements are complete and agreed in similar projects or domains (or project).
		FEF 3.1.3 adapt/ spread changes in domain requirements to concrete projects within domain.	When the requirements are changing in the requirements repository, the changes are spread to all related requirements within the domain.
		FEF 3.1.4 provide comparison of domain requirements feasibility.	The RE-tool has to support functionality for requirements analyses in the requirements repository by comparing, changing and storing them here.
	FEF 3.2. Generate predefined and ad hoc reports, documents that comply with standard industrial templates, with support for presentation-quality output and in-built document quality controls.	**FEF 3.2.1** provide wizards for report generation.	Wizards help to prepare reports, which are not designed together with the RE-tool functionality.
		FEF 3.2.2 provide possibility to print report according to views and sorting.	The RE-tool has to maintain requirements sorting, filtering, prioritisation requirements agreement, negotiation and rationale maintenance reports which would be possible to view, send by e-mail, and print them out.
		FEF 3.2.3 provide possibility to print results of rationale, brainstorm and etc.	
		FEF 3.2.4 provide techniques for error checking.	All the reports have to be syntactically correct. The RE-tool should have functionality for error prevention, detection and correction.
	FEF 3.3. Generate the complete specification, expressed using formal language (informal and semiformal languages might also be included), commonly agreed by all stakeholders.	**FEF 3.3.1** correspond to standards of software documentation.	The requirements specification should correspond to the international standards (the exact standard maintained by the RE-tool, is determined by the non-functional process requirements).
		FEF 3.3.2 correspond to standards, defined by an organisation.	The RE-tool should have means to define internal organisational standards for requirements specification. The internal requirements specification standards could be adapted following the international ones, or created according the organisational needs.
		FEF 3.3.3 support formal languages for complete, commonly agreed requirements specification.	The output of the RE process is an agreed, complete, represented in formal language requirements specification. The RE-tool has to ensure functionality for completeness, agreement and formality check.

Figure VI-40 - Specification dimension

3.2. Framework for Non-functional RE-tool Requirements

Process Requirements

Le détail des fonctionnalités du framework se trouve dans la Figure VI-41.

Product Requirements

Le détail des mesures du framework se trouve dans la Figure VI-42.

External Organisational Requirements

Le détail des mesures du framework se trouve dans la Figure VI-43.

External Requirements for Business Parties

Le détail des mesures du framework se trouve dans la Figure VI-44.

	Feature RE-tool should…	Working process characteristics
Process requirements	**NF1.1.** Support the selected RE and requirements specification standards.	IEEE std 830-1998, ISO 9126, NASA-DID-P200, Internal organisational standards.
	NF1.2. Support the selected modelling perspectives.	Structural, functional, behavioural, rule, object, communication, actor and role.
	NF1.3. Support the software development models.	Waterfall, spiral, prototyping, transformational, knowledge-based, domain-based, and RUP.
	NF1.4. Support the interfaces with the editing, modelling and implementation tools.	Editing, communication, modelling, implementation, testing tools, database management systems.

Figure VI-41 - Non-functional Process Requirements

	Feature	Measures	Questions
Product requirements	**NF2.1.** Provide a user satisfying usability of the system.	Capability of the system to be: – understood; – learned; – used and liked by the user; – used under specified conditions.	How easy is the RE-tool to learn for various groups of users? How efficient is the RE-tool for the frequent user? How easy is the RE-tool to remember for the occasional user? How satisfied is the user with the RE-tool? How easy is it to understand what the RE-tool does?
	NF2.2. Provide a user satisfying reliability of the system.	Accuracy Error tolerance; Consistency; Recoverability; Availability.	What failure types are evaluated for the RE-tool? What is the availability of the RE-tool and its components? How long is the RE-tool allowed to be out of operation after it has failed?
	NF2.3. Provide a user satisfying performance of the system.	System performance; Required amount of resources; Data and result accuracy.	What is the minimum and average response time for certain data transaction or operation? What is minimum and maximum throughput for certain operations? What is the capacity of the RE-tool? What is resource utilization of the RE-tool? Do users receive the exact output?
	NF2.4. Provide a user satisfying supportability of the system.	Correctability Extensibility	What type (e.g., corrective, adaptive, preventive, perfective) of maintainability is performed most often? What are reasons for the maintainability activities? How many steps (activities, corrections) should be performed to reach the desirable state of the tool?

Figure VI-42 - Non-functional Product Requirements

	Features	Measures	Description
Organisational requirements	**NF3.1.1.** Decide about the biggest costs the organisation is able to pay for the RE-tool evaluation?	Tool prices	Direct tool price, tool running costs (like new computers, RAM, etc.)
		Evaluation costs	Evaluation and transition costs (changing from the old to a new working practice), time and resources need to evaluate tool.
		Maintenance and support costs	Future operational costs, support costs, maintenance/ upgrades etc.
	NF3.1.2. Evaluate the decision making approach in the organisation.	Objectivistic (management requirements);	An objectivistic world-view describes an organisation, where stakeholders can map the required RE-tool without changing the organisational processes.
		Constructivistic (users' requirements)	A constructivistic approach supports stakeholder knowledge externalisation and internalisation processes, and in such a way makes the organisational environment part of the individual understanding. The RE-tool selection is the process of negotiation of the environment and RE-tool suitability to this environment.
	NF3.1.3. Investigate organisational experience of the software usage.	Degree of the RE-tool requirements	Different user experience contributes with different degree of abstraction for the tool functional and non-functional feature evaluation. More experienced users could require evaluating bigger number of requirements or only the particular requirements, which would contribute with evaluation of 'needed' tool features.
		Degree of RE-tool evaluation needed	Different user experience contributes with different degree of abstraction for the tool functional and non-functional feature evaluation. More experienced users could require the more detailed tool analysis of a particular feature.
	NF3.1.4. Define the legitimate constraints social norms.	Political constraints and laws	Political constraints and laws specify the norms, which exist in the organisation, and the country where organisation is situated.
		Cultural constraints	Cultural constraints describe the national traditions, customers, which could effect tool evaluation and usage.
		Social constraints	Social constraints characterise the organisational workers relationships, organisational workers hierarchy and relationships between management and workers.

Figure VI-43 - Non-functional External Organisational Requirements

	Features	Measures	Description
Requirements for business	**NF3.2.1.** What facilities the business parties are suggesting for user training?	Availability (costs) of training seminars;	Describes availability of tool training seminars organised by the tool vendor.
		Time spent on the phone before getting an answer;	Does the vendor or consultant provide phone, e-mail and online consultancy services using messenger systems (e.g.: MSN, ICQ and Skype). How long does it take to get answers about the tool usability and functionality?
		Average response time for online help;	
		Average response time for help via e-mail;	
		Availability of information in the vendors' Web page.	Does the vendor of the tool provide the appropriate support in the tool Web page? Is the information informative and relative to the emerging question? How often does the vendor update the Web information?
	NF3.2.2. Do the business parties provide the adequate system maintenance?	Quality of the maintainability services;	The basic supportability (maintainability) metrics: ratio of total change implementation time to total; number of changes implemented; number of unresolved problems; time spent on unresolved problems; percentage of changes that introduce new faults; number of modules modified to implement a change.
		Availability at any time;	Is vendor available at any time when maintainability problems of the tool emerge? How long does it take to get service form the vendor?
	NF3.2.3. Do the business parties provide the adequate system support?	Quality of support during new version releases;	Does vendor representative participate during tool updates? Does the vendor provide teaching services of new tool versions?
		Quality of updates of the current tool version	Do updates solve the old version problems? Are new tool versions more effective, easy to use, etc. than older ones?

Figure VI-44 - Non-functional External Requirements for Business Parties

Annexe XVI. Seilevel Requirements Tool Evaluation

3.1. Outils concernés par l'étude d'exemple

1. IBM Rational DOORS

ADRESSE WEB :	http://www-01.ibm.com/software/awdtools/doors/productline/

2. Siemens Teamcenter

ADRESSE WEB :	http://www.plm.automation.siemens.com/en_us/products/teamcenter/

3. Blueprint Requirements Center

ADRESSE WEB :	http://www.blueprintsys.com/

4. eDevTECH inteGREAT Requirements Studio

ADRESSE WEB :	http://www.edevtech.com/requirements_studio.html

5. IBM Rational Composer

ADRESSE WEB :	http://www-01.ibm.com/software/awdtools/rrc/

6. 3SL Cradle

ADRESSE WEB :	http://www.threesl.com/

7. Microsoft Team Foundation Server

ADRESSE WEB :	http://www.microsoft.com/visualstudio/en-us/products/2010-editions/team-foundation-server/overview

8. Jama Software Contour

ADRESSE WEB :	http://www.jamasoftware.com/contour/

9. Polarion Requirements

ADRESSE WEB :	http://www.polarion.com/products/requirements/index.php

10. HP Quality Center

ADRESSE WEB : http://www8.hp.com/us/en/software/software-product.html?compURI=tcm:245-937045-&pageTitle=quality-center

11. Orcanos Qpack

ADRESSE WEB : http://www.orcanos.com/Orcanos_QPack.htm

12. TraceCloud

ADRESSE WEB : http://www.tracecloud.com/GloreeJava2/jsp/WebSite/TCHome.jsp

13. Sparx Systems Enterprise Architect

ADRESSE WEB : http://www.sparxsystems.com/products/ea/index.html

14. Kovair Application Lifecycle Management

ADRESSE WEB : http://www.kovair.com/alm/index.aspx

15. TechnoSolutions TopTeam Analyst

ADRESSE WEB : http://www.technosolutions.com/topteam_requirements_management.html

16. MKS Integrity

ADRESSE WEB : http://www.mks.com/platform/our-product

17. Micro Focus Caliber RM/RM

ADRESSE WEB : http://www.microfocus.com/products/caliber/caliberrm/index.aspx

3.2. Fonctionnalités visées par l'étude

BA adds a bulk of new requirements at once

- Bulk enter requirements in the tool directly (RF - it's not entirely clear how this one is different than the above 2, would this be like copying and pasting a bunch of requirements from say word into the tool without explicit 'import' functionality?
- Automatically identify requirements from external text document by key words, structure, etc. (e.g. specify what keywords to search for in the doc and import based on those)
- Batch import structured data as new requirements from Excel (e.g., import Excel file of previously developed requirements into the new managed system)
- Batch import data as new requirements from Word (e.g., import Word file of previously developed requirements into the new managed system)

BA adds new requirements individually

- Add new requirement
- Allows a unique ID assignment to each requirement or object (could be a manual entry of an ID)
- Automatically check requirements for grammar usage
- Automatically check requirements for spelling errors
- Automatically check requirements for use of ambiguous words
- Automatically create unique ID for each requirement
- New requirements entered are marked as 'new' for review
- Support entering tables within requirements
- Enter individual requirements with minimal user interaction (e.g., mouse clicks, keyboard presses, etc. - in Word, you can press 'Enter' to enter another requirement)
- Document a requirement with rich text formatting

BA creates open issues for requirements

- Allow issue creation from the requirement itself and automatically link the issue to the requirement
- Automated integration to issue tracking tool (preferably Sharepoint)
- Requirement can have a link to an issue in another tool as a URL (e.g. a link to the issue in SP)
- Tool has issue tracking as part of its functionality

BA does not want a requirement anymore

- Delete requirements
- Allow a delete option on requirements that doesn't forever lose the requirement

BA ensures requirements are implemented correctly

- Assign verification methods to requirements
- Auto generate a draft UAT script for a set of requirements
- Create a regression test suite based on the requirements
- Create a system test suite based on the requirements
- Create UAT test scripts linked to requirements
- Link requirement to status of product verification (from requirement the user can see support test plans, determining test plan and test procedure coverage, determination of successful test)
- Automated integration with test management system (e.g. QualityCenter)

BA exports requirements for review outside the tool

- Create an output from the tool that you can review it and make changes in the output that get put back into the tool
- Define custom format for documentation output (e.g., provide an SRS template that the system populates)
- Export a subset of requirements based on a filter view applied
- Export requirements to delimited format (e.g., CSV)
- Export requirements to HTML
- Export requirements to Word (doc or rtf)
- Export requirements to XML
- Highlight changes on exported document against a previous baseline
- Preview of requirements document generation
- Produce requirements documentation in various pre-defined military and commercial standard formats.

BA is offline and needs to continue work on the requirements

- Automatically synchronizes when coming back online
- Provide ability to work disconnected (i.e., no connection to requirements repository) and merge changes upon reconnecting (may require user to trigger)
- Periodically reminds user to synchronize offline work2

BA is offline and wants to reference something in the requirements

- Provide ability to work disconnected in read-only view

BA is reviewing requirements

- Filter for a list of new/changed requirements and review the changes
- Mark a requirement as a duplicate of another and merge them together
- Step through process flows with mapped screen shots and see the relevant screens
- Requires only a few clicks to view detailed information about requirements (Like excel not a web-based tool where it's buried many pages deep)
- Merge changes between work from different analysts on a set of requirements
- In lists of requirements, can scan the list and see old versions of the requirement in the list. (For example, the client may have a certain way of saying something and I may restate it. As we go down the list of requirements it is helpful to see the current correct version, but also see the original version to jog their memory)
- Quickly read through many requirements with minimal user interaction (e.g., mouse clicks, keyboard presses, etc.)
- Restrict two people from concurrently editing a requirement in online mode

BA is trying to find a requirement or set of them

- Search requirements by keyword

BA is trying to find gaps in requirements

- Create links between requirements of the same type
- Display traceability results in a diagram (e.g., a tree)
- Navigate traceability hierarchy easily (e.g. I am looking for missing requirements so I start with a high level requirement and want to see the low level requirements that flow from it, I then want to check the next high level requirement and check the low level requirements that flow from it.)
- Traceability analysis to identify missing links within the requirements (e.g., functional requirement orphans not linked to a use case)
- Display any kind of traceability results in a table
- Display customized view of traceability data in a table where user can select which types of objects show up in the columns

BA links existing documentation to requirements

- Automated integration with revision control system for any linked documents - (e.g., SharePoint, CVS, SourceSafe, etc.)
- Link client documentation to one or more requirements
- Link requirements to actual documents in a SharePoint location (e.g. I can click it and it opens)
- Store a URL to a document in a SharePoint location (or other network path)
- Link requirements to documents that are not stored on a local drive (e.g., store linked document in a database)
- Can link video files related to requirements to one or more requirements
- Can upload and store video files about requirements in the tool
- Can link audio files related to requirements to one or more requirements
- Can upload and store audio files about requirements in the tool

BA makes updates to requirements

- Edit individual requirements quickly without having to go through multiple clicks to get to an editable view (like editing in XL vs. many layers on a web-interface)
- Edit requirements
- Edit requirements with undo functionality
- Notify affected project participants by e-mail about requirement changes
- View requirements as a list to make it easy to scan many requirements at once and change fields directly in that list
- Bulk edit properties, meaning you can select multiple requirements and edit a common property and all items would receive that value (e.g., select 20 requirements and bulk update each of their status to Approved)
- Edits are updated to the master copy of the data in real-time (e.g., DB vs. a doc)
- When copy/paste into fields from Word or Excel, Tool maintains existing field formatting

BA models requirements

- Automatically generate diagrams from written requirements (e.g., context diagram, use case diagrams)
- Describe a requirement with an embedded document in the database (e.g., a Visio diagram)
- Incorporate UI requirements with a UI prototype (e.g., define a set of requirements or an image and associate it to the requirements. Let you click/hover to view those requirements)
- Incorporate UI requirements with specific sections of a UI prototype (e.g., define areas of an image that have associated requirements. Let you click/hover to view those requirements)
- Link flows to steps in other flows
- Link models to sub-objects in other models (link decision tree to a step in a process flow)
- Link requirements to a full model
- Link requirements to sub-objects in models (link requirements to a step of a process flow, to a system in a context diagram, to a state object in a state diagram)
- Map screenshots to process flows or use case steps
- Model context diagrams directly in the tool

- Model org charts directly in the tool
- Trace requirements to existing Visio documents
- Visually can model a use case and it automatically translates it into text requirements
- Describe a requirement with an image directly in the database (e.g., a bitmap context diagram or Visio file image)
- Model process flows directly in the tool

BA navigates to find requirements

- Navigate the requirements hierarchy by going up and down and across branches
- Pull up a list of requirements associated with a given model (example is all requirements for a process flow or all requirements for a UI screen)
- Show the requirements in a hierarchy
- For an individual requirement, can visually emphasize where in the requirements hierarchy it sits

BA needs approval on requirements

- Define 1 custom workflow for all types of requirements
- Define custom workflow for each type of requirement
- Pre-defined workflows setup already
- Request requirements approval/signoff via email
- Track requirements approval/signoff

BA queries for requirements data that isn't available in standard reports

- Perform ad-hoc queries of requirement data
- Support open database system (standard query access)

BA resolves open issues on requirements

- Tracks open issues by requirement
- Tracks resolutions to open issues on requirements

BA reviews requirements with customer

- Group requirement sets for review based on attributes (e.g. when changed) and allow the set to be accepted or rejected. (e.g. a set that can be published for review all at once, people then would accept the whole set or not. If it gets rejected I can make the fixes, or remove offending requirements and resubmit the change request for approval. Once it is approved everyone who has registered to be notified, is notified of changes.)
- Make requirements modifications in a list quickly and in real-time as we have a discussion (e.g. on a projector in a meeting make live edits)
- Make requirement attribute modifications quickly as we have a discussion (e.g. in a requirements review session and need to make notes and reassign people as we go.)

BA takes notes in session about requirements

- Link notes (created in another tool) from gathering sessions to individual requirements
- Tool allows OneNote integration

BA wants help on how to use the tool

- Availability of context-sensitive help within the system
- Availability of documentation (either online, soft-copy, or hard-copy)
- Availability of learning aids (e.g., sample requirements, workflow tutorial, etc.)
- Availability of online support (e.g., message board, knowledge base, users group, wiki, etc.)
- Availability of phone support
- Availability of training classes

BA wants to capture email context for requirements

- Link emails to specific requirements (Someone will send me a request via email and I will make the change. I would like to link the email to the requirements so that I can easily refer to the request the person sent.)
- Link email to a specific version of a requirement in the change history (Someone will send me a request via email and I will make the change. I would like to link the email somehow to the change so that I can easily refer to the request the person sent.)

BA wants to see requirements linked to other requirements

- Create links between defined groups of requirements
- Create links between many requirements all at once (e.g. preselect a set and link them all individually to another requirement in a mass update)
- Drag and drop to link requirements to one another (e.g. want to map my requirements to use cases very quickly. I believe that drag and drop to link requirements would be fast)
- Generate traceability reports (downward relationship from a given set of requirements, upward relationship from a given requirement
- Provide drag-and-drop ability to move requirements within the hierarchy.
- Report to show any problems with requirement links (e.g., parent requirement is invalid so children are marked as invalid).
- Describe the nature of the relationship link between requirements (whether it is a "traces from", "dependency", etc.)
- Create links between requirements of different types

BA wants to track the origin of requirements

- Capture source of requirements
- Capture statements of rationale associated with requirements statements and collections
- Track who originally requested a specific requirement

BA wants to understand what changes were made to a requirement

- Automatically maintain audit trail for requirement changes (user, time/date, annotation of change, and change detail)
- History of requirements changes is easy to view

BA wants to use tool to capture multiple types of information about requirements

- Define and capture different types of requirements (e.g., use cases, functional requirements, etc.)
- Define a glossary for a set of requirements.
- Defined glossary for a set of requirements allows you to hover or click on keywords to get the definition

Business prioritizes requirements

- Captures what release requirement is slated for
- Link requirements to those of another requirement type such as business objectives
- Set priority on requirements
- Voting on requirements

Business reviews requirements and makes comments or edits

- Track conversations about requirements, attach conversations to requirement objects

Developers use requirements to develop code

- Automated integration with design tools (e.g., Rose, XDE, Enterprise Architect, etc.)
- Automated integration with development environments (e.g., Eclipse)

IT enhances functionality of the RM tool

- External API available

Look at a set of requirements to take action on a group of them

- Filter a view of requirements by criteria (e.g., view all requirements in Draft status)
- Filtered view applies instantly (It needs to happen fast because I am holding things in my short term memory that I will forget if it takes too long)
- Save private custom views (e.g., analyst makes custom filtered view and plans to use often in the future)
- Save public custom views (e.g., manager makes a custom view that other analysts will also use)
- Sort view of requirements by multiple criteria at once (e.g., view all requirements in ascending order by creation date)
- Apply filters to custom attributes of the requirements objects in addition to out of the box attributes (I have a set of requirements that can be functionally divided (i.e. tax/shipping/etc) but can also be divided by region (US, EMEA, APAC) and want to be able to view the requirements in the way that is convenient to me.)

Manager assigns requirements work to different people

- Assign different requirements groups to different analysts for completion (e.g. owners on requirements or sets of requirements)

Manager creates new reports

- Customize reports to include any attributes of the requirements objects

Manager setups up permissions to access requirements

- Manage access permissions for groups of users
- Manage access permissions for individual users (e.g., some users can edit requirements, others can only read them)
- Restrict requirements from clients based on some attribute of the requirements (e.g. only let them see requirements in a non-draft status)
- Restrict sets of requirements from clients based on project they belong to (e.g. only let them see the requirements for their project)

Manager wants to quickly skim status on requirements

- Customize dashboard reports
- View dashboard reports

Manager wants to see how much work is left on the requirements

- Create burndown reports on requirements status
- Report on the maturity of requirements by nature of requirements status and number of reviews per requirement
- Report on the number of reviews of requirements or sets of requirements
- Report on whether a set of requirements have been reviewed (looked at)

Manager wants to see open issues on requirements

- Report on number of issues open, closed, in progress, etc. about requirements by requirement or some other attribute
- Report on all current open issues about requirements

Manager wants to see status of requirements work

- Export data from any report to Excel (e.g., export metrics data to Excel for further analysis)
- Metrics charting (e.g., color graph of volatility data over time)
- Metrics reporting (e.g., progress/status, requirements velocity, requirement changes, tracing metrics, etc.)

Manager wants to see who owns requirements

- Report on who is working on what requirements
- New BA has to get up to speed on tool
- Easy to learn
- Instinctually easy to use

Requirements architect wants to setup information management structure for the project

- Can store fluffy text around the requirements to give people context as to what the purpose of a given set of requirements are or other types of explanation
- Define a stakeholder for each requirement
- Define custom format for requirements IDs (e.g., UCXX for use cases, BRXX for business rules, etc.)
- Define dependencies between requirements separate from hierarchy of requirements
- Define hierarchical relationships between requirements
- Define what data should be captured for each type of requirement (i.e., create custom data fields, specify if the information is required/optional, etc.)
- Group requirements by project (i.e., system supports multiple projects)
- Organize requirements by groups (e.g., group by sub-system)
- Set whether requirement attributes are required or not
- Setup complex attributes with predefined value choices (e.g. custom status values)
- Provide a template for new projects (e.g., includes pre-defined requirement types) to minimize setup
- Can add different types of requirements

BA rolls back/Reviews a prior set of requirements

- Compare baselines of the requirements
- Create baselines of the requirements
- Need to be able to baseline so we can show the difference, and what is new between the last version and this version
- Reproduce an earlier version of requirements exactly as they were
- Work with specific baseline of the requirements
- Roll back to a prior version of a requirement